Universidad Zaragoza
1542

Trabajo Fin de Máster

Evaluación de modelos de aprendizaje profundo mediante redes neuronales guiadas por datos para materiales no lineales.

Evaluation of Deep Learning models through data-driven neural networks for nonlinear materials.

Autor

Javier Orera Echeverría

Directores

Jacobo Ayensa Jiménez

Manuel Doblaré Castellano

ESCUELA DE INGENIERÍA Y ARQUITECTURA

2021

Index

Abstract

Nonlinear materials are often difficult to model with classical methods like the Finite Element Method, have a complex and sometimes inaccurate physical and mathematical description or simply we do not know how to describe such materials in terms of relations between external and internal variables. In many disciplines, neural network methods have arisen as powerful tools to deal with nonlinear problems. In this work, the very recently developed concept of Physically-Guided Neural Networks with Internal Variables (PGNNIV) is applied for nonlinear materials, providing us with a tool to add physically meaningful constraints to deep neural networks from a model-free perspective. These latter outperform classical simulation methods in terms of computational power for the evaluation of the prediction of external and specially internal variables, since they are less computationally intensive and easily scalable. Furthermore, in comparison with classical neural networks, they filter numerical noise, have faster convergence, are less data demanding and can have improved extrapolation capacity. In addition, as they are not based on conventional parametric models (model-free character), a reduction in the time required to develop material models is achieved compared to the use of methods such as Finite Elements. In this work, it is shown that the same PGNNIV is capable of achieving good results in the predictions regardless of the nature of the elastic material considered (linear, with hardening or softening behavior), being able to unravel the constitutive law of the material and explain its nature. The results show that PGNNIV is a useful tool to deal with the problems of solid mechanics, both from the point of view of predicting the response to new load situations, and to explain the behavior of materials, placing the method in what is known as Explainable Artificial Intelligence (XAI).

Resumen

Los materiales no lineales son normalmente difíciles de modelar con métodos clásicos como los Elementos Finitos, tienen una descripción física y matemática compleja y en ocasiones inexacta o simplemente se desconoce cómo modelarlos mediante relaciones entre variables internas y externas. En muchas disciplinas, las redes neuronales han surgido como una herramienta muy poderosa para tratar problemas no lineales. En este trabajo, el concepto recientemente desarrollado de Redes Neuronales Guiadas por Física con Variables Internas (PGNNIV, *Physically-Guided Neural Networks with Internal Variables*) se aplica a materiales no lineales, proporcionando una herramienta que permite añadir restricciones con sentido físico a redes neuronales profundas, evitando la necesidad de establecer modelos paramétricos. Estas últimas son mejores que los métodos de simulación clásica en la predicción de variables externas y especialmente internas, pues son computacionalmente menos intensivas y fácilmente escalables. Adicionalmente, en comparación con las redes neuronales clásicas, las PGNNIV filtran el ruido numérico, tienen una convergencia más rápida, requieren menos cantidades de datos y tienen una mayor capacidad de extrapolación. Además, al no estar basadas en modelos paramétricos convencionales, se consigue una reducción del tiempo necesario para desarrollar modelos de materiales en comparación con el uso de métodos como los Elementos Finitos. En este trabajo, se demuestra que una misma PGNNIV es capaz de arrojar buenos resultados en las predicciones independientemente de la naturaleza del material elástico considerado (lineal, con endurecimiento o con reblandecimiento), siendo capaz además de desentrañar la ley constitutiva del material, explicando su naturaleza. Los resultados demuestran que las PGNNIV resultan una herramienta muy útil para tratar los problemas de la mecánica de sólidos, tanto desde el punto de vista de la predicción de la respuesta a nuevas situaciones de carga, como para explicar el comportamiento de los materiales, situando el método en lo que se conoce como Inteligencia Artificial Explicable (XAI).

Chapter 1

Introduction

1.1. The Big Data Paradigm

It is of common knowledge that our everyday life is dramatically influenced by Big Data or Data Analytics. These methodologies allow the processing of data that would be intractable by traditional data/software applications. They are also capable of unraveling hidden information behind the data, and most importantly, behind unstructured data [1, 2]. This new approach to data has also influenced the way we conceive science, specially through the concept of Data-Driven Models. The so-called Internet of Things gives us, furthermore, billions of data in many specific fields such as preventive maintenance, and leads nowadays to the development of innovative and efficient systems aiming at increasing operational efficiency in a new generation of smart factories [3].

Some scientists even speculate that, given the impressive development of Big Data and computing, theorethical physics might also be tractable by its tremendous power. A recent article published in The New York Times [4] claims that "the Theory of Everything is still not in sight, but with computers taking over many of the chores in life — translating languages, recognizing faces, driving cars, recommending whom to date — it is not so crazy to imagine them taking over from the Hawkings and the Einsteins of the world."

However, the paucity of data when trying to solve some scientific problems has strongly determined the pathway of this new developed approach. Furthermore, all state variables

in physical problems must be known for a given measure as their dependence is the one that will be inferred from measurements. This is not always the case in practical situations, being this one of the main concerns in Data Science [5]. Anyway, the power to extract knowledge from unstructured data is obvious.

Artificial Intelligence has risen nowadays as a powerful tool in science and technology to extract hidden and complex patterns and make predictions from data by means of Machine Learning methods. Machine Learning techniques split into two different branches: **supervised learning**, when both inputs and their desired outputs (labels) are known and the system learns to map inputs to outputs, and **unsupervised learning** when true outputs are not known and the system itself discovers the structure patterns within the data [6]. Classification and regression (including conventional linear and polynomial regression and Support Vector Regression [7]) are examples of supervised learning while data clustering (e.g. k-means algorithm [8]) and dimensionality reduction (e.g. principal components analysis [9]) are examples of non-supervised learning.

Manifold Learning is a clear example of how to extract complex and abstract patterns from unstructured data. This sub-field of machine learning operates in continuous domains and learns from observations that are represented as points in a certain space [10]. The goal of such learning is to discover the underlying relations between observations, on the assumption that they lie in a limited part of the space, typically a manifold, the intrinsic dimensionality of a manifold of which is an indication of the degrees of freedom of the underlying system [11]. Among the different existing techniques that follow this Data-Driven approach, we find kernel Principal Component Analysis (kPCA), Self Organizing Map (SOM), Locally Linear Embedding (LLE), Isomap, Laplacian Eigenmap, t-distributed Stochastic or Neighbor Embedding (t-SNE) [11].

A particular framework for both supervised and unsupervised learning is Artificial Neural Networks (ANN). These can be used for both classification and regression [6]. Neural networks are nowadays black boxes that, when well trained, have yielded extraordinary results in many fields like image and speech recognition, natural language processing (specially some architectures of Convolutional Neural Networks (CNN)) [12], social and economic models and many other fields of science. In the field of image recognition, there exist big databases and numerous neural network configurations that have been developed surpassing the classification power of techniques such as Bayesian

classification, Nearest neighbor, SVM and decision trees. Examples of that are some CNN networks such as LeNet (1998), GoogleLeNet (2015) or ResNet (2015) [13, 14]. The empirical results obtained reveal the superiority of neural networks over statistical techniques such as exponential smoothing or auto-regressive integrated moving average among others [15].

1.2. Data-Driven methods and Physical Systems

Data driven methods are used in many different physical disciplines such as chemical and electrical processes [16], biology [17], spoken language recognition [18] and a long etcetera. However, the link between physics and data-driven methods has not been clear. In [19] new forms of empiricism that declare "the end of theory" and the creation of data-driven methods rather than knowledge-driven science are explored.

This novel approach arises in this data-driven context, with the main purpose of turning the apparently not-physically meaningful data-driven models into physics-aware models. We present a brief overview of the contributions made in this research line in the last years along different fields of science.

Partial Differential Equations (PDEs) have a extremely strong power to model physical phenomena. Analytic solutions are rarely easy to find and, in most cases, they do not exist globally. That is the reason why numerical methods have become the universal tool to give approximate but accurate solutions of PDE. However, Data Science and, in particular Machine Learning tools, seemed to be separated from the aforementioned PDE approach during the last decades. In the last ten years, attempts to solve PDE from a data-driven point of view have been numerous. For instance, [20] presents a deep learning-based approach that can handle general high-dimensional parabolic PDEs. The machine learning of more and more physical laws is therefore a consequence of the work carried out on the symbiosis between PDEs and deep learning. For example, [21] develops a method to learn physical systems from data that fulfills the first and second principles of thermodynamics.

In the field of continuum mechanics, this methodological approach is still not fully developed, but its potential foresees that powerful tools can be built on an interaction between continuum mechanics and machine learning, (specially deep learning and dynamic networks) and its applications in fields such as health-care or medical diagnosis.

In order to achieve the objective of providing these models with a meaningful physical character, many efforts have been done to introduce physical information between the input and output layers of the network. In [22], an energy informed machine learning method to approach a computational mechanics problem is applied to several sub-fields such as linear elasticity, elastodynamics, hyperlasticity and phase field modeling of fracture or Kirchhoff plate bending. In [23] the potential of solving inverse problems with linear and non-linear behavior is tackled using deep learning methods. In particular, the forward problems are solved first to create a database, which is then used to train the machine learning algorithms and determine the boundary conditions of a problem from assumed measurements. Other examples of problems related with continuum mechanics, problems such as the solution of the Navier-Stokes equations have been investigated and encoded in a physics informed deep learning framework [24].

However, in all these works, there is not a clear mathematical framework for associating physical laws to the particular value of neurons at the hidden layers thus not solving the problem of the *black box* character of NN and therefore not providing explanatory capacity to the models.

1.3. The problem of the internal variables

The differentiation between external and internal variables becomes an important factor when referring to the new Big Data approach. External variables are those observable, measurable variables of the system, that can be obtained directly from physical sensors such as position, temperature or force sensors; internal variables are non-observable (not directly measurable) variables, that integrate locally other observable magnitudes and depend on the particular internal structure of the system [25]. In all the presented works, the involved fields are treated similarly, regardless of whether they are internal variables and therefore not measurable. The question that now arises is whether it is possible or not to develop new data-based models to solve those physical problems and incorporate in a proper manner internal variables to the computations.

As explained before, most of physical systems are described in terms of partial differential equations (PDEs) that are currently computationally solved by means of numerical methods. These methods consider a given discretization in space and time which results on an algebraic, in general non-linear system, that is then solved by means of standard matrix manipulation. Until now, Data-Driven approaches to the problem have either not whitened the black box [26–28] or need explicitly a definition of the

cloud of experimental values that identifies the internal state model [29]. The latter methods correspond to the so-called Physically-Guided Data Science (PGDS), whose direct application are the Data-Driven Simulation-Based Engineering and Sciences (DDSBES) [5, 30].

The perspective followed in this work, however, describes and develops another Machine-Learning approach coined as Physically-Guided Neural Networks with Internal Variables (PGNNIV) [31]. This new methodological approach arises in the aforementioned context and provides a tool to predict internal variables by physically constraining the hidden layers of a deep neural network using the fundamental laws of the universe. Thanks to the deep learning modeling power and the constraints that physically meaningful equations provide, not only the real internal variables of a physical problem are predicted, but also the data needed to train the network decreases, convergence is reached faster, external variable predictions reach higher filtering accuracy and it is even possible to extrapolate out of the ranges of the training dataset, as it has been demonstrated recently [25, 31].

Based on these assumptions, this thesis aims to export these ideas to a particular field as it is Nonlinear Solid Mechanics. In [31], the theoretical basis of the application of the PGNNIV methodology is discussed and some numerical examples on the field of fluid mechanics (flow inside of a pipe) are proposed and verified. The application of this methodology to continuum problems is discussed in [25]. Here we present a specific framework for the problem we are interested in.

1.4. Main objectives

In this thesis we tackle the possibility of extending the PGNNIV concept to solid mechanics with infinitesimal strains, in particular to linear and nonlinear elastic and hyperelastic materials. The main objective of this work is to adapt a previous model already implemented for linear homogeneous materials to nonlinear materials. With this aim, we increase the complexity of the networks by adding layers and neurons, and a new developed *deconvolutional* technique is used for the prediction and identification of internal variables such as strains and stresses. We proceed under the umbrella of the PGNNIV methodology, already developed and tested. Therefore, it is not our objective to demonstrate the advantages of PGNNIV when compared to other numerical methods (this has been already demonstrated [31]), but to adapt it and to ensure its validity in a new topic as solid mechanics.

Particularly, we have set up the subsequent specific objectives:

1. **Analysis of the performance of the linear PGNNIV topology on linear materials:** The performance of linear PGNNIV topology on linear materials is analyzed.

2. **Development of a procedure to automatically generate variable data-sets with a finite element software**: This procedure allows any user to easily run any number of simulations and create, therefore, data-sets as big as desired and with a customized variability, including any type of material and boundary conditions (applied loads) by just changing a few parameters in the code.

3. **Development of a new nonlinear PGNNIV topology for simplified nonlinear elastic materials (softening and hardening materials):** This objective poses the major challenge of the thesis. After having created the desired nonlinear material data-set, a PGNNIV master must be designed, its hyper-parameters adjusted and the network trained. A single network should be able to capture the behavior of any nonlinear elastic material under the assumption of infinitesimal strain theory, including both hardening and softening behavior.

4. **Development of a new nonlinear PGNNIV topology for real and general nonlinear hyperelastic materials (neohookean and test-based materials):** Provided that a successful implementation of the nonlinear neural network topology is achieved on nonlinear elastic materials, we aim to use this topology to be able to predict the behavior of an extended family of real hyperelastic materials.

5. **Comparison of the linear and nonlinear PGNNIV topologies for all materials tackled:** We do not only want to prove that the developed PGNNIV topology can learn and predict nonlinear elastic and hyperelastic external and internal variables, but also that it achieves better results than the former topology conceived for linear materials.

1.5. Structure of the document

The content in this document is structured in five chapters. Firstly, a brief introduction presenting the state of the art is included together with the main objectives and goals if this work. In Chapter 2, a description of the theoretical framework and the methods employed to create the neural network models are presented. In Chapter 3, the main results for linear and nonlinear elastic materials are presented, and in Chapter 4 we extend the methodology to real hyperelastic materials. Finally, in Chapter 5 we summarize and discuss the results extracted and we give a brief overview of the future work on finite strains that is already being developed.

Chapter 2

Theoretical background and methods

In this chapter we present the theoretical framework of the thesis. For that, we summarize solid mechanics infinitesimal strain theory and present the approach followed for the use PGNNIV in this context.

2.1. Theoretical framework.

2.1.1. Brief summary of infinitesimal solid mechanics.

In this thesis we work in the context of Newtons' Mechanics, and assuming that solids are continuum and the principles of thermodynamics apply. The main assumptions of infinitesimal solid mechanics are [32, 33]:

1. **Small displacements**: Equations can be written in the deformed configuration because changes between deformed and not deformed configurations are extremely small.

$$\boldsymbol{x} \simeq \boldsymbol{X} \Rightarrow \boldsymbol{u}(\boldsymbol{X}, t) \simeq \boldsymbol{u}(\boldsymbol{x}, t) \tag{2.1}$$

2. **Small deformations**: Displacement derivatives are extremely small, as well as their products.

$$\left| \frac{\partial u_i}{\partial x_j} \right| \ll 1 \tag{2.2}$$

$$\frac{\partial u_i}{\partial x_j}\frac{\partial u_i}{\partial x_j} \ll \frac{\partial u_i}{\partial x_j} \tag{2.3}$$

where u is the displacement field and $i, j \in \{1, 2, 3\}$.

Also, we shall assume [34]:

3. **Static problem**: Loads are applied in a quasi-static way.

$$\ddot{\boldsymbol{u}} \simeq \mathbf{0} \tag{2.4}$$

4. **Isothermal problem**: No temperature change is considered.

$$T(\boldsymbol{x}, t) = T(\boldsymbol{x}) \tag{2.5}$$

The two fundamental variables under these hypotheses are the stresses and the strains. On each point of a solid we can define the strain and stress tensor, denoted $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ respectively, and also called Cauchy stress tensor and Cauchy strain tensor.

The Cauchy stress tensor σ_{ij} is defined as a tensor that links the stress in the current configuration defined by its normal vector n_j:

$$t_i^n = \sigma_{ji} n_j \tag{2.6}$$

where t_i^n is the stress vector in a plane with normal n_j.

Starting from Newton's first law, it is possible to formulate the equilibrium equations in terms of the Cauchy stress tensor [32, 33] as:

$$\sigma_{ji,j} + \rho b_i = 0 \tag{2.7}$$

where b_i is the external force vector per unit volume and ρ is the density of the material.

On the other hand, taking into account that the fulfillment of the equation of angular momentum, assuming negligible external distributed momenta, we obtain that the Cauchy stress tensor is symmetric:

$$\sigma_{ij} = \sigma_{ji} \tag{2.8}$$

The Cauchy deformation tensor ε_{ij} can be defined in terms of the displacement field as:

$$\varepsilon_{ij} = \frac{1}{2}(u_{i,j} + u_{j,i}) \tag{2.9}$$

where $\boldsymbol{u}$ is the displacement field.

Given a tensor ε_{ij}, it corresponds to a displacement field if and only if such tensor field satisfies the so-called compatibility conditions, that may be written as follows [32]:

$$\nabla \times (\boldsymbol{\varepsilon} \times \nabla) = \mathbf{0} \tag{2.10}$$

Or in index notation:

$$\varepsilon_{ij,kl} + \varepsilon_{kl,ij} - \varepsilon_{ik,jl} - \varepsilon_{jl,ik} = 0 \tag{2.11}$$

For the problem we tackle, we make use of the equations of infinitesimal elasticity, which can be put together as a whole with its correspondent boundary conditions as follows:

1. Stress equilibrium in the domain:

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} + \rho \boldsymbol{b} = 0 \tag{2.12}$$

2. Strain Cauchy strain tensor as a function of the displacements:

$$\boldsymbol{\varepsilon} = \frac{1}{2}(\nabla \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \nabla) \tag{2.13}$$

3. Equilibrium on the boundary:

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} = \bar{\boldsymbol{t}} \text{ in } \Gamma_N \tag{2.14}$$

   where $\boldsymbol{n}$ is the normal unitary vector to the boundary surface and $\bar{\boldsymbol{t}}$ is the external boundary traction vector in the boundary Γ_N.

4. Compatibility on the boundary.

$$\boldsymbol{u} = \bar{\boldsymbol{u}} \text{ in } \Gamma_D \tag{2.15}$$

   where $\bar{\boldsymbol{u}}$ is the displacement vector in a region of the boundary Γ_D.

2.1.2. The constitutive state equation.

Kinematic and dynamic equations for the Solid Mechanics are able to link displacements with strains and external forces with stresses respectively. However, they can not separately link the displacements with the external forces for a deformable solid [32, 33]. Therefore and in order to close the problem, it is necessary to introduce the material behavior, which is, based on our experimental observations, different for each material (they yield a different displacement field under the same external forces). For the most general material, the constitutive law that contains the material information would be of the form [32, 33]:

$$\boldsymbol{\sigma} = \boldsymbol{f}(\boldsymbol{\varepsilon}, T, t, \boldsymbol{q}, \boldsymbol{X}, \boldsymbol{\kappa}) \tag{2.16}$$

where T is the temperature of the solid, t is the time, $\boldsymbol{q}$ is a set of internal variabes that account for the history, $\boldsymbol{X}$ is the position and $\boldsymbol{\kappa}$ are some parameters particular to the material itself.

We are going to assume the following simplifications:

1. $\boldsymbol{f}$ is not history-dependent.

2. Quasi-static loading, therefore $\boldsymbol{f}$ is time-independent.

3. $\boldsymbol{f}$ is independent on the position $\boldsymbol{X}$, so the material is homogeneous. Homogeneous materials have the same constitutive law for every point of the domain. On the contrary, for heterogeneous materials, the constitutive equation differs from one point of the solid to another.

4. $\boldsymbol{f}$ is independent on the temperature T.

Therefore, we can establish the homogeneous anisotropic material model:

$$\boldsymbol{\sigma} = \boldsymbol{f}(\boldsymbol{\varepsilon}, \boldsymbol{\kappa}) \tag{2.17}$$

Taking into account that the material parameters $\boldsymbol{\kappa}$ are fixed for a given material, and following an index notation, the constitutive law of elastic or inelastic materials under the infinitesimal strain theory is therefore of the form:

$$\sigma_{ij} = \boldsymbol{f}(\varepsilon_{ij}) \tag{2.18}$$

where $\boldsymbol{f}$ can have in principle of any functional shape, even if in most cases, we assume some extra conditions about the nature of $\boldsymbol{f}$. For example, nonlinear elastic materials such as hyperelastic materials are described by means of a strain energy function $\Psi = \Psi(\varepsilon_{ij})$ as follows:

$$\sigma_{ij} = \frac{\partial \Psi}{\partial \varepsilon_{ij}} \tag{2.19}$$

In particular, the constitutive law of a homogeneous **linear** elastic material writes as:

$$\boldsymbol{\sigma} = \mathbf{C} : \boldsymbol{\varepsilon} \tag{2.20}$$

Figure 2.1, extracted from [35], shows this linear relation:

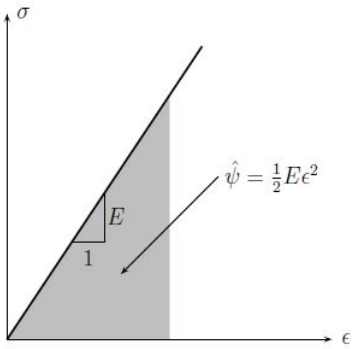


Figure 2.1: Stress-strain curve for a linear elastic material subject to uni-axial stress [35].

This equation, also known as the generalized Hooke's Law, defines the most general linear relation among all the components of the stress and strain tensors [35], which for a general linear homogeneous anisotropic material (material parameters $\boldsymbol{\kappa}$ are different along different directions) reads, using index notation:

$$\sigma_{ij} = C_{ijkl}\varepsilon_{ij} \tag{2.21}$$

where C_{ijkl} are the components of the fourth-order stiffness tensor of material properties or *Elastic moduli*.

Due to the symmetries of $\boldsymbol{\sigma}$ and $\boldsymbol{\varepsilon}$ and to the definition of $\boldsymbol{\sigma}$ in terms of the strain energy density Ψ [35], the fourth-order stiffness tensor has 21 independent components for three-dimensional problems [35], if the material is anisotropic. We can make further simplifications depending on the extra symmetries of the material:

1. **Orthotropic material**: It has two or three orthogonal axes, along which properties are different. Therefore, C_{ijkl} has 9 independent coefficients in the 3D space. This matrix links the components of the strain and stress tensors, that is, $(\varepsilon_{xx}, \varepsilon_{yy}, \gamma_{xy})^T$ and $(\sigma_{xx}, \sigma_{yy}, \sigma_{xy})^T$, and for planar stress reads:

$$\boldsymbol{C} = \begin{pmatrix} \frac{E_1}{1-\nu_{12}\nu_{21}} & \nu_{21}\frac{E_1}{1-\nu_{12}\nu_{21}} & 0 \\ \nu_{12}\frac{E_2}{(1-\nu_{12}\nu_{21})} & \frac{E_2}{1-\nu_2\nu_{21}} & 0 \\ 0 & 0 & G_{12} \end{pmatrix} \tag{2.22}$$

   where E_1 and E_2 are the elastic moduli in x and y directions respectively, ν_{12} and ν_{21} are the Poisson's ratio in both directions and we have defined the shear modulus $G_{12} = \frac{E_1}{2(1+\nu_{12})}$. Besides, it must be fulfilled that $\nu_{12}E_2 = \nu_{21}E_1$. Therefore we have only 3 independent parameters for planar stress.

2. **Transversely isotropic**: The physical properties are symmetric about an axis that is normal to a plane of isotropy [35]. Therefore, C_{ijkl} has 5 independent coefficients in the 3D space.

3. **Isotropic material**: The constitutive law does not depend on the considered direction. Therefore, C_{ijkl} has 2 independent coefficients in the 3D space. The matrix links the components of the strain and stress tensors, that is, $(\varepsilon_{xx}, \varepsilon_{yy}, \gamma_{xy})^T$ and $(\sigma_{xx}, \sigma_{yy}, \sigma_{xy})^T$, and for planar stress reads:

$$\boldsymbol{C} = \frac{E}{(1-\nu^2)} \begin{pmatrix} 1 & \nu & 0 \\ \nu & 1 & 0 \\ 0 & 0 & \frac{1}{2}(1-\nu) \end{pmatrix} \tag{2.23}$$

   For that case, the shear modulus is $G = \frac{E}{2(1+\nu)}$.

It is common to have access to the results of a certain test performed to a given material. The path is therefore to fit these test-based materials to a certain (probably nonlinear) functional form for the strain energy density Ψ, assuming therefore some dependence $\boldsymbol{\sigma} = \boldsymbol{f}(\boldsymbol{\varepsilon})$.

In this thesis we tackle linear and nonlinear elastic materials (Chapter 3) as well as strain-energy-function based hyperelastic and test-based materials, both using a defined strain energy function and fitting a strain energy function function to data (Chapter 4) under the infinitesimal strain theory. The PGNNIV paradigm proves to be able to explain all of them with a single network topology.

2.1.3. Numerical Methods.

Numerical methods are mathematical algorithms to solve mathematical problems. They are written in the language of mathematics and nowadays implemented as numerical algorithms. Thanks to the boost of both theoretical computer science and the development of powerful software tools, numerical methods have determined the successful progress of almost any area of science and engineering, from fields such as signal processing to quantum, fluid and solid mechanics. The discretization of either space and time domains is the basis of numerical methods.

In the particular case of solid mechanics under the hypotheses here considered, time discretization becomes not relevant, since loads are applied in a quasi-static way, and the sole discretization of the geometry provides a very good approximation of how continuum solids behave.

The traditional finite element methods used in the physical problem of elastic solids use an matrix-based approach to get algebraic systems to find an accurate solution. By subdividing the whole domain into small parts (elements) the partial differential equation governing the physical phenomena occurring on the particular geometry can be solved. These PDEs are approximated by computable functions to generate algebraic systems for complex geometries. Although these methods have many advantages (they can discretize complex geometries, with a variety of solving algorithms for numerous engineering and scientific applications), they require to exactly know the properties of the material, and are time-consuming when aiming at solving the problem for many different load-cases.

On the contrary, the new PGNNIV paradigm requires no information about the material properties since these are learned during the training process of the network and reduces the calculation time to seconds when the network is already trained.

For the discretization of the problem the 2D-discretized divergence and the 2D-discretized symmetric gradient are used. The symmetric gradient operator is defined as:

$$\text{SGRAD}(\boldsymbol{u}) = \frac{1}{2}\left(\nabla \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \nabla\right) \tag{2.24}$$

where $\boldsymbol{u}$ is the displacements vector. The divergence of a tensor $\boldsymbol{\sigma}$ is defined as:

$$\text{DIV}(\boldsymbol{\sigma}) = \boldsymbol{\nabla} \cdot \boldsymbol{\sigma} \tag{2.25}$$

These two differential operators are framed as artificial neural network convolutional filters:

1. In order to discretize the symmetric gradient, we first have to take derivatives of the displacements $\boldsymbol{u}$, which is a vectorial field. We need to approximate derivatives on each element e^{ij}, where $(i,\, j)$ is the indexation of a pixel in the 2D-solid. For example, the derivative of the displacement $u^{ij}_{x,y}$ is computed as $\frac{1}{2}\left(\frac{u_x^{i+1j+1} - u_x^{i+1j-1}}{\Delta h_y} + \frac{u_x^{i-1j+1} - u_x^{i-1j-1}}{\Delta h_y}\right)$ where Δh_y is the differentiation step in y direction.

2. Similarly, in order to discretize the divergence of a tensor, we also discretize the derivatives in the element e^{ij}. For example: $\sigma^{ij}_{x,x}$ is computed as $\frac{1}{2}\left(\frac{\sigma_x^{i+1j+1} - \sigma_x^{i-1j+1}}{\Delta h_x} + \frac{\sigma_x^{i+1j-1} - \sigma_x^{i-1j-1}}{\Delta h_x}\right)$ where Δh_x is the differentiation in x direction.

2.1.4. Mechanical and geometrical set-up of the case study: uniaxial and biaxial test on a rectangular board.

The case study considered in this work consists of a biaxial test on a rectangular board with height 16 cm and width 20 cm. For that, we consider a certain compression load profile $p = p(s)$ (where s is the coordinate along the right and top contour), which is symmetric with respect to the X and Y axis acting perpendicular to the board contour, as shown in Figure 2.2:

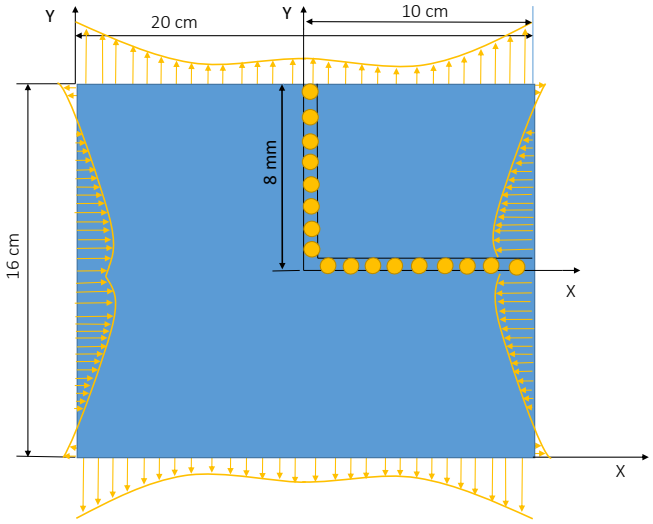


Figure 2.2: Dimensions and representation of the biaxial test on the board

No volumetric loads are considered, that is, $\boldsymbol{b} = \mathbf{0}$. As shown in Figure 2.2, for simplicity and to accelerate the calculation processes, symmetry allows us to analyze an equivalent problem by extracting a quarter of the whole board and applying, as boundary conditions, no movement on the bottom face in the Y direction and no movement of the left face in the X direction. Sliding of both faces along the boundary is of course permitted, as shown in Figure 2.2.

2.2. Data-Driven approach of Solid Mechanics

2.2.1. PGNNIV at a glance.

Classical deep neural networks are black boxes that theoretically can compute and learn any kind of function [36]. In particular, they perform very well in many areas of science and technology. Although there exist some heuristic rules, these black boxes are usually trained via trial and error. Adding a physical meaning to the hidden layers, and constraining them by adding an extra term to the cost function we want to minimize, has already proven to yield very satisfactory results such as few data requirements, higher accuracy and faster convergence speed in real physical problems [31]. The basic principles of PGNNIV and it comparison to classical networks are briefly exposed in the next lines.

PGNNIV are in principle a generalization of what is called Physics Informed Neural Networks (PINN) [37]. In these latter the physical equations constrain the values of

output variables to belong to a certain physical manifold. PGNNIV outperform PINN since the physical equations constrain the values in some intermediate layers so that the network has an explanatory capacity.

Let us consider a set of continuous partial differential equations of the form:

$$\begin{aligned} \mathcal{F}(u, v) &= f, \text{ in } \Omega \\ \mathcal{G}(u) &= g, \text{ in } \partial\Omega \\ \mathcal{H}(u) &= v, \text{ in } \Omega \end{aligned} \tag{2.26}$$

where u and v are the unknown fields of the problem, $\mathcal{F}$ and $\mathcal{H}$ are functionals representing the known and unknown physical equations of the specific problem, $\mathcal{G}$ is a functional that specifies the boundary conditions, and f and g are known fields.

The continuous problem has its analogous discretized representation in finite-dimensional spaces in terms of vectorial functions $\boldsymbol{F}$, $\boldsymbol{G}$ and $\boldsymbol{H}$ and nodal values $\boldsymbol{u}$, $\boldsymbol{v}$, $\boldsymbol{f}$ and $\boldsymbol{g}$. Particularly, $\boldsymbol{u}$ are the solution field nodal values and $\boldsymbol{v}$ are the unknown internal field variables at the different nodes or elements. A PGNNIV may be defined for a problem of the type (2.26) in the following terms:

$$\begin{aligned} \boldsymbol{y} &= \mathsf{Y}(\boldsymbol{x}) \\ \boldsymbol{v} &= \mathsf{H}(\boldsymbol{u}) \\ \boldsymbol{x} &= \boldsymbol{I}(\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{g}) \\ \boldsymbol{y} &= \boldsymbol{O}(\boldsymbol{u}, \boldsymbol{f}, \boldsymbol{g}) \\ \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{v}, \boldsymbol{f}, \boldsymbol{g}) &= 0 \end{aligned} \tag{2.27}$$

where:

1. $\boldsymbol{R}$ are the physical constraints, related to the relations given by $\boldsymbol{F}$ and $\boldsymbol{G}$.

2. $\boldsymbol{I}$ and $\boldsymbol{O}$ are functions that compute the input $\boldsymbol{x}$ and the output $\boldsymbol{y}$ of the problem, that is, the data used as starting point to make predictions and the data that we want to predict.

3. Y and H are models. Y is the **predictive model**, whose aim is to infer accurate values for the output variables for a certain input set and H is the **explanatory model**, whose objective is to unravel the hidden physics of the relation $\boldsymbol{u} \rightarrow \boldsymbol{v}$. These two models are defined at the PGNNIV framework as some neural networks with a specified topology.

The following graphical representation gives an overview of the aforementioned methodology:

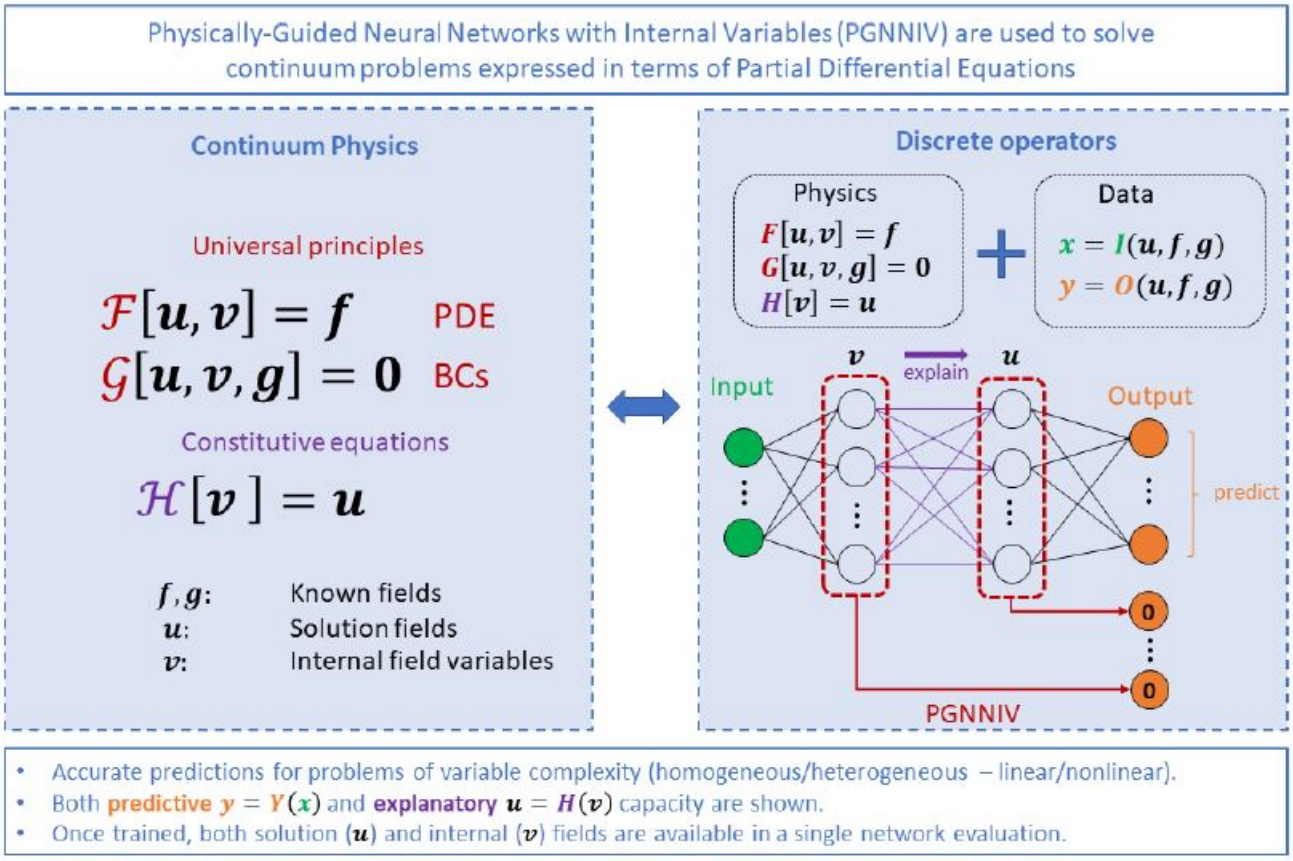


Figure 2.3: Schematic representation of the methodology considered. Figure extracted from [31].

The appropriate structure and topology for H and Y depends on the problem in hands and is discussed later in this work.

2.2.2. PGNNIV for Solid Mechanics.

In this section we go in depth in the analysis of the data-driven model for our case study.

The data that contains the nodal and element-wise variables (that is, displacements, and stresses and strains respectively) is stored in high dimensional structures since our problem is two-dimensional in space and planar stress behavior is considered. These structures are 1D-, 2D- and 3D-images with a number of pixels that correspond to the dimensions of the board. An image containing the information of a variable $\boldsymbol{I}$ can be written as $\boldsymbol{I}_k^{ij}$, where $k = \{x, y, xx, yy, xy, \ldots\}$ is the tensor component of the variable in a 2D-Cartesian space, and i and j are the spatial indices of the pixel considered. For example $\boldsymbol{u}_x^{ij}$ corresponds to the value of the displacement in X-direction in the pixel (i, j) of the board and σ_{xy}^{ij} corresponds to the value of the shear stress XY in the pixel (i, j) of the board. The neural network training process uses batches and not single images. Therefore, these 2D- and 3D-images are stored as batches and one more dimension is added to consider samples variability. This is the common tensor-based framework used by Tensorflow.

Figure 2.4 shows a graphical representation of the different images and links between them. We have an input tensor, that is, forces $\boldsymbol{f}_x^r$, $\boldsymbol{f}_y^t$, (where r and t mean right and top boundary respectively), on certain elements of the boundary, and we have an output, that is, displacements $\boldsymbol{u}_x$,$\boldsymbol{u}_y$ on each pixel. The conventional NN approach consists of a neural network connecting the mentioned input and output. However, the results obtained using this approach show no information at all about the internal variables and lack explanatory capacity.

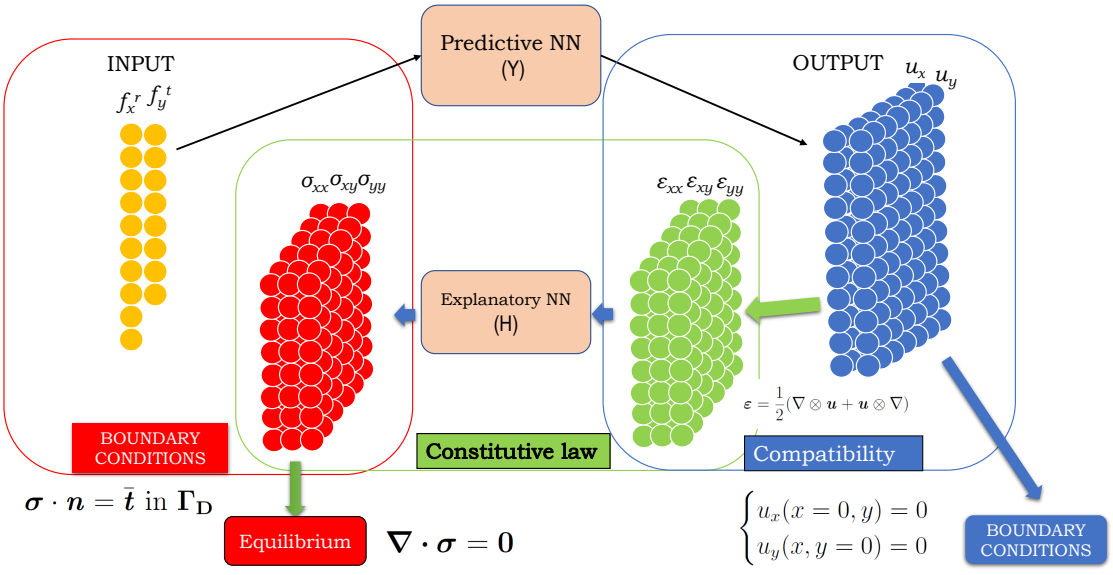


Figure 2.4: Graphical representation of the PGNNIV in 2D planar stress conditions on the board.

This is the point were PGNNIV become a powerful tool to overcome these limitations. Let us now tackle the PGNNIV on the specific continuum mechanics problem. First, we had a data-driven model (NN) that correlates the applied forces with the resulting displacements. If we now add some internal layers that are physically meaningful, we can actually denote them as real internal variables, that is, stresses and strains. Stresses have to fulfill the equilibrium condition in the domain and boundaries. On the other hand, strains have to be defined in terms of the displacements using the definition of the strain tensor in terms of $\boldsymbol{u}$ and the displacements are prescribed at the boundary (compatibility on the boundary). The neural network relating stresses and strains has now a physical meaning, since it builds the constitutive material law itself. Now we have two models working as a whole, that means, the weights and biases that they contain mutually converge to the desired solution. Figure 2.4 gives an overview of PGNNIV used for the solid mechanics problem.

It is important to remark that these two models, which are both neural networks, contain the information about the material and the influence of external forces on each displacement pixel; they can be considered as one single model with two sub-models that work in parallel. Additionally, if we are under the linear elasticity constrains, we know that we can work with linear NN, that is, NN with no internal layers and no activation functions (only a weights matrix). However, if we want to let the model learn the physical behavior of a nonlinear material, a more complicated model with hidden layers (in particular, a deep neural network) and specific algorithms are needed. From now on we identify a linear neural network as a set of weights and biases with no activation functions in-between, and a nonlinear neural network denotes a proper deep neural network.

Classical neural network for the case study. Defining the input variable *load* as an external stimuli $\boldsymbol{f} = \boldsymbol{x_0}$, each hidden layer of n_i neurons, $\boldsymbol{x}_i$, $i = 1, \cdots, L$ is defined with a functional relation:

$$\boldsymbol{x}_i = \phi(\boldsymbol{x}_{i-1}\boldsymbol{W}_i + \boldsymbol{b}_i) \tag{2.28}$$

where $\boldsymbol{W}_i$ and $\boldsymbol{b}_i$, $i = 1, \cdots, L$ (L is the number of layers) are weights and biases, which are the parameters of the model, and $\phi : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is an activation function. Given a set of *ground truth* data points $\mathcal{E} = \{\boldsymbol{f}^i, \bar{\boldsymbol{u}}^i | i = 1, \cdots, N\}$, where $\boldsymbol{f}^i$ corresponds to

the load stimuli applied to the board (uniaxial or biaxial test) and $\bar{\boldsymbol{u}}^i$ to the resulting displacements measured by some kind of procedure or sensors (although in this work, these data is generated synthetically as we detail later), predicted as $\boldsymbol{u} = \boldsymbol{x}_L$ in the case of the output layer. The cost function CF used to train the network is the mean squared error and has the following form:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} ||\bar{\boldsymbol{u}}^i - \mathsf{Y}(\boldsymbol{f}^i; \boldsymbol{W})||^2 \tag{2.29}$$

where $\bar{\boldsymbol{u}}^i$ represents the observed nodal displacement corresponding to sample i, $\boldsymbol{f}^i$ represents the load stimuli applied to the board corresponding to sample i, and $\mathsf{Y}(\boldsymbol{f}^i; \boldsymbol{W})$ represents the network, depending on some parameters (weights and biases) $\boldsymbol{W}$.

Adding sub-networks and constraints to obtain the PGNNIV. Identifying the general problematic (presented in the equation system in (2.27)), each discrete variable, operator and model of the problem has a corresponding one in the Solid Mechanics field. This means that Solid Mechanics can be intrinsically considered from the PGNNIV point of view. We identify our problem as a predictive and explanatory problem with internal hidden variables and constraints. Indeed, for our problem we have:

$$\begin{aligned} \boldsymbol{u} &= \mathsf{Y}(\boldsymbol{f}) \\ \boldsymbol{\sigma} &= \mathsf{H}(\boldsymbol{\varepsilon}) \\ \boldsymbol{x} = \boldsymbol{I}(\boldsymbol{u}, \boldsymbol{f}) &= \boldsymbol{f} \\ \boldsymbol{y} = \boldsymbol{O}(\boldsymbol{u}, \boldsymbol{f}) &= \boldsymbol{u} \\ \boldsymbol{R}(\boldsymbol{u}, \boldsymbol{\varepsilon}, \boldsymbol{\sigma}, \boldsymbol{f}) &= [\text{DIV}(\boldsymbol{\sigma}) = \boldsymbol{0}, \boldsymbol{\sigma} \cdot \boldsymbol{n} = \boldsymbol{f}|_{\Gamma_N}] \end{aligned} \tag{2.30}$$

where $\boldsymbol{u}$ is the nodal displacement field, $\boldsymbol{f}$ is the nodal applied load field, $\boldsymbol{\sigma}$ is the elemental stress tensor and $\boldsymbol{\varepsilon} = \text{SGRAD}(\boldsymbol{u})$ is the elemental strain tensor.

The problem (2.30) is solved using a PGNNIV where the loss function includes a term related with the error between the predictions and true values of the output variable and other penalty terms related to some (physically-based) equations, that are, equilibrium and strain-displacements constraints. Getting into the details, the different terms are:

1. Loss term associated to the measurement of the displacement field:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} ||\bar{\boldsymbol{u}}^i - \mathsf{Y}(\boldsymbol{f}^i)||^2 \tag{2.31}$$

where $\bar{\boldsymbol{u}}^i$ is the observed displacement corresponding to sample i.

2. Constraint associated with the equilibrium equation.

$$\boldsymbol{\nabla} \cdot \boldsymbol{\sigma} = \mathbf{0} \tag{2.32}$$

3. Constraint associated with the strain-displacement relation.

$$\boldsymbol{\varepsilon} - \frac{1}{2}(\nabla \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \nabla) = \mathbf{0} \tag{2.33}$$

4. Constraint associated with the compatibility of the displacements on the boundary.

$$\begin{cases} u_x(x = 0, y) = 0 \\ u_y(x, y = 0) = 0 \end{cases}$$

using the coordinate system imposed in Figure 2.2.

5. Constraint associated with the equilibrium of the stresses on the boundary.

$$\boldsymbol{\sigma} \cdot \boldsymbol{n} - \boldsymbol{f} = \mathbf{0} \text{ in } \Gamma_N \tag{2.34}$$

All in all, the global cost function can be represented as OF=MSE+PEN, with PEN referring to the terms 2 to 5 (3 is identically $\mathbf{0}$ by definition):

$$\begin{aligned} \text{OF} = p_1 \frac{1}{N} \sum_{i=1}^{N} ||\bar{\boldsymbol{u}}^i - \Upsilon(\boldsymbol{f}^i)||^2 + p_2 \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{\nabla} \cdot \boldsymbol{\sigma}||^2 + \\ p_3 \frac{1}{N} \sum_{i=1}^{N} ||\boldsymbol{\sigma} \cdot \boldsymbol{n} - \boldsymbol{f}||^2 + p_4 \frac{1}{N} \sum_{i=1}^{N} ||u_x(x = 0, y) + u_y(x, y = 0)||^2 \end{aligned} \tag{2.35}$$

where p_i are penalty coefficients, that compute the relative importance of each term in the global OF. Recall that no penalty for the compatibility in the domain is included since $\boldsymbol{\varepsilon} - \frac{1}{2}(\nabla \otimes \boldsymbol{u} + \boldsymbol{u} \otimes \nabla)$ is indentically $\mathbf{0}$.

The minimization problem writes therefore as:

$$\min_{\boldsymbol{W}} \text{OF}(\mathcal{E}, \boldsymbol{W}) \tag{2.36}$$

where $\boldsymbol{W}$ are the network parameters and $\mathcal{E} = \{\boldsymbol{f}^i, \bar{\boldsymbol{u}}^i | i = 1, \cdots, N\}$ is a given training data-set. By minimizing this function we obtain the desired results in terms of displacement, stresses and strains.

2.3. Data and Software

2.3.1. Data generation

Firstly, it is important to mention that we generate the data artificially since we did not collect data from a real system. Artificial generation is also faster and cheaper in this case where an accurate numerical solution is available, faster and cheaper.

The data-set corresponding to the linear material was generated with a finite element script in MATLAB software environment. The units used are Pa for the elastic modulus definition, cm for the displacements and geometry and 10^{-4} N for the nodal forces.

For the nonlinear material creation, the software used was the following: Abaqus, as our finite element software, to simulate different load-cases with different materials; MATLAB in co-simulation with Abaqus CAE for the iterative creation of artificial data-sets with enough variability.

The artificial data-sets corresponding to nonlinear materials used to train the network were generated using Abaqus CAE/6.14-2 in co-simulation with MATLAB. Appendix A contains the code that iteratively runs a Abaqus simulation applying a different load case each simulation. Firstly, we use Abaqus2Matlab [38] to link both tools. This tool automatically creates a Matlab code that can interact with the command window from the computer and extract the chosen variable fields. The variability in the data-set is achieved by randomly changing the load profile, so that the Abaqus input file generated on each simulation is different. The units used to define the magnitudes in the case study of this work are MPa for pressure values, and cm for length values.

The board model contains 10×8 elements, to which we equally call *pixel*, since they are considered as values of a displacement, stress or strain *image*. For these last two variables four values per pixel were generated, one for each integration point of each element. A variable mean value for each element is extracted afterwards.

Within the Abaqus input file environment we are able to activate or disable the option

for large deformations `NLGEOM`. Normally, when we use the infinitesimal strain theory framework we disable it. However, hyperelastic or test-based materials require from its activation since its behavior is considered by the solver under the finite strain theory. Nevertheless, the value of the boundary conditions considered keep the solid behavior within the infinitesimal strain theory domain, that is, regions where strains are sufficiently small.

The load profiles acting on the board right and top contours are parabolic:

$$\begin{aligned} f^r &= (a_0 + a_1 y + a_2 y^2) \\ f^t &= (b_0 + b_1 x + b_2 x^2) \end{aligned} \tag{2.37}$$

where f_i is the force actuating in the i direction (x or y direction) perpendicular to the board surface and r and t refer to the right boundary and top boundary respectively. The coefficients are calculated so that the value at the points within the range $x = [-5.5, 5.5]$, $y = [-4, 4]$ is approximately between 0 and 1, although in some cases these values may differ.

The load profile that is used as input for the board actuating external pressure or forces is generated as a parabolic profile with parameters P_i, which are the values of the load at the center and at the corners of the boundaries, as illustrated in Figure 2.5. These values vary from 0 to 1 MPa for the nonlinear material and 0 to 1 [10^{-4} N] for the linear material.

The coefficients a_i and b_i are obtained as follows, based on P_i parameters and the geometry.

$$f(0) = P_1 \Rightarrow a_0 = P_0 \tag{2.38}$$

$$f\Big(\frac{c}{2}\Big) = P_2 \Rightarrow P_2 = P_0 + a_1\frac{c}{2} + a_2\frac{c^2}{4} \tag{2.39}$$

$$f\Big(\frac{-c}{2}\Big) = P_1 \Rightarrow P_1 = P_0 - a_1\frac{c}{2} + a_2\frac{c^2}{4} \tag{2.40}$$

Subtracting (2.3.1)-(2.40) we obtain:

$$a_1 = \frac{P_2 - P_1}{c} \tag{2.41}$$

Solving for a_2 in equation (2.40)

$$a_2 = 2\frac{P_2 + P_1 - 2P_0}{c^2} \tag{2.42}$$

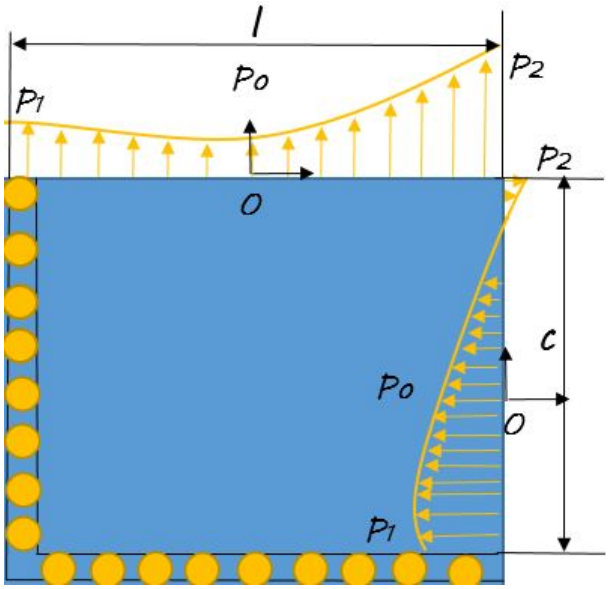


Figure 2.5: Board load profile parameters.

The values of P_0, P_1 and P_2 are generated randomly between 0 to 1 MPa for the nonlinear material and 0 to 1 [10^{-4} N] for the linear material. The definition of the right load profile equation coefficients according to these parameters writes therefore as follows:

$$\begin{aligned} a_0 &= P_0 \\ a_1 &= \frac{P_2 - P_1}{c} \\ a_2 &= 2\frac{P_2 + P_1 - 2P_0}{c^2} \end{aligned} \tag{2.43}$$

The definition of the top load profile is analogously obtained:

$$\begin{aligned} b_0 &= P_0 \\ b_1 &= \frac{P_2 - P_1}{l} \\ b_2 &= 2\frac{P_2 + P_1 - 2P_0}{l^2} \end{aligned} \tag{2.44}$$

However, in many occasions the top load profile coefficients are different, using c instead of l:

$$\begin{aligned} b_0 &= P_0 \\ b_1 &= \frac{P_2 - P_1}{c} \\ b_2 &= 2\frac{P_2 + P_1 - 2P_0}{c^2} \end{aligned} \tag{2.45}$$

Therefore, the load profiles do not strictly vary from 0 to 1 MPa.

2.3.2. Network creation and training

Nowadays, there is a large number of neural network software available in the field of machine learning, and existing softwares with other purposes such as MALTAB have developed tools to deal with neural network models. Among the top artificial neural networks libraries we find Tensorflow, PyTorch or Keras.

In particular, Tensorflow provides us with the flexibility to create our own cost functions, as well as serving as a tool to build our own tensors and networks architectures. It is an extensive library that makes apparently complex tensor operations tractable. Furthermore, it provides an ideal framework to deal with neural networks. Tools such as the backpropagation algorithm, automatic differentiation, as well as complex optimization algorithms, activation functions and several neural network topologies such as convolutional layers are presented in a user-friendly manner.

For all the neural network models to be presented throughout the thesis the Adam optimizer is used. Adams optimizer implements the Adam algorithm stochastic gradient descent method based on a adaptive estimation of the first-order and second-order moments [39]. Empirical results demonstrate that Adam works well in practice and compares favorably to other stochastic optimization methods [40].

Besides the optimizer and the specific network topology, several hyper-parameters have to be chosen to design the network. These are the learning rate, also known as β, which is a input parameter for the optimizer and determines the optimization step and the penalty coefficients p_i of Equation (2.35).

Finally, the interaction and visualization of results is carried out by means of MATLAB, and since the framework in which the Tensorflow software is implemented is Python, we can easily link the results to MATLAB.

Chapter 3

Linear, softening and hardening materials

In this chapter we present the results obtained when a linear material is modeled with a linear PGNNIV, explain the methodology followed to create a nonlinear PGNNIV for softening and hardening materials and compare the predictions of both PGNNIV (linear and nonlinear) when working with these materials.

3.1. Linear Elasticity

In this section we analyze PGNNIV in the context of linear elasticity. We present the results obtained for a linear elastic material with a linear PGNNIV topology. For that, we calculate the prediction errors of the parameters of the material for the isotropic, orthotropic and anisotropic cases and extract conclusions from the capability of predicting the desired parameters. Our aim is to ensure that the PGNNIV framework is appropriate for the mechanical problem in order to make sure that it can be extended to nonlinear materials.

3.1.1. Constitutive modeling of linear materials using PGNNIV

The starting point is the already well-developed PGNNIV approach for a linear elastic material.

Following the notation employed at Equation(2.27), we have defined:

$$\begin{aligned} \boldsymbol{u} &= \mathsf{Y}(\boldsymbol{f}) \\ \boldsymbol{\sigma} &= \mathsf{H}(\boldsymbol{\varepsilon}) \end{aligned} \tag{3.1}$$

where Y is a the **predictive** network and H is the **explanatory** network representing the constitutive model.

Since we work under the framework of linear elasticity, networks Y and H have to be able to reproduce linear relations, so they have neither hidden layers nor activation functions, that is, there are simply linear functions.

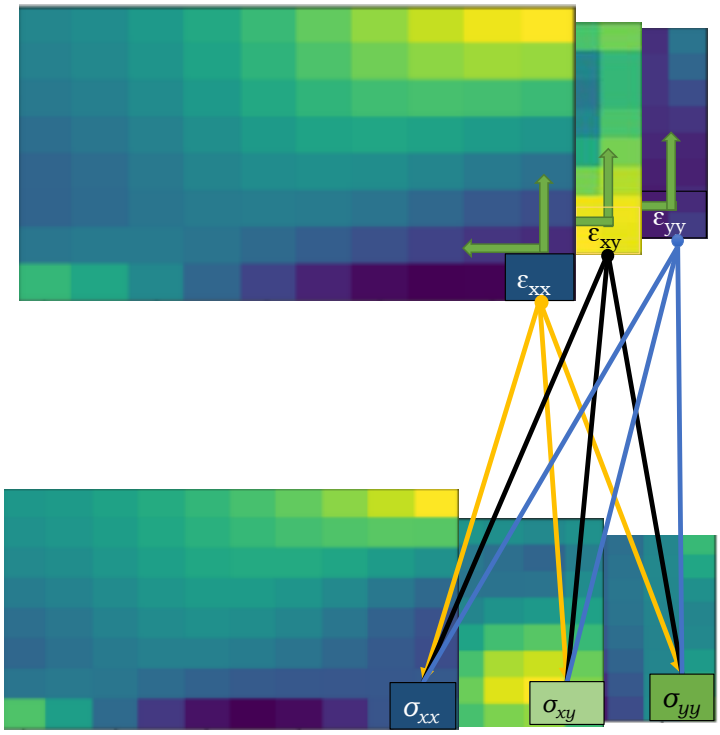


Figure 3.1: Graphical representation of the linear network topology Y model

In Figure 3.1 we illustrate the network topology associating the values of the strains at one given element to the values of the stresses on the same element, that is, the network H.

As we can recall from section 2.1.1 (Equation (2.23)), the symmetric stiffness matrix that defines the **isotropic** material is defined as follows: Defining the strain and stress

state as the vectors, $(\varepsilon_{xx}, \gamma_{xy}, \varepsilon_{yy})^T$ and $(\sigma_{xx}, \gamma_{xy}, \sigma_{yy})^T$, we have:

$$\boldsymbol{C} = \frac{E}{(1-\nu^2)} \begin{pmatrix} 1 & 0 & \nu \\ 0 & \frac{1}{2}(1-\nu) & 0 \\ \nu & 0 & 1 \end{pmatrix} \tag{3.2}$$

the matrix built to predict both parameters is of the form:

$$\hat{\boldsymbol{C}} = \begin{pmatrix} a & 0 & b \\ 0 & \frac{1}{2}(a-b) & 0 \\ b & 0 & a \end{pmatrix} \tag{3.3}$$

where a and b are the network parameters.

We directly obtain the two constitutive parameters from the predicted matrix as:

$$\begin{aligned} \nu &= b/a \\ E &= (a-b)(1+b/a) \end{aligned} \tag{3.4}$$

For the **orthotropic** material we define a matrix containing 3 different constants for the plane stress stiffness matrix (see Equation (2.22)). We expect to predict the 5 material parameters: $E_1, E_2, \nu_{12}, \nu_{21}, G_{12}$ of the stiffness matrix in terms of 3 constants a, b and c (we know some elements are 0 due to symmetry), and it is of the form:

$$\hat{\boldsymbol{C}} = \begin{pmatrix} a & 0 & b \\ 0 & c & 0 \\ b & 0 & d \end{pmatrix} \tag{3.5}$$

We directly obtain both parameters from the predicted matrix as:

$$\begin{aligned} \nu_{21} &= b/a \\ \nu_{12} &= b/d \\ E_1 &= a - b/d^2 \\ E_2 &= d - b^2/a \\ E &= \frac{E_1 + E_2}{2} \\ \nu &= \frac{\nu_{12} + \nu_{21}}{2} \end{aligned} \tag{3.6}$$

An isotropic material will contain parameters satisfying $E_1 = E_2$, ν_{12}=ν_{21} and G_{12} given by:

$$c = G_{12} = \frac{E_1}{2(1+\nu_{12})} \tag{3.7}$$

Analogously, for the anisotropic material (containing 6 independent parameters for plane stress), we make sure all the components fulfill the relations with the two unique variables of the real isotropic material: E and ν. The network must predict all components of the matrix that, taking symmetries into account, make a total of 6 independent parameters:

$$\hat{\mathbf{C}} = \begin{pmatrix} a & e & d \\ e & b & f \\ d & f & c \end{pmatrix} \tag{3.8}$$

For the anisotropic material we directly obtain all parameters from the predicted matrix the same way as for the isotropic material using Equations (3.4). We also check that e and f are close to zero.

3.1.2. Results of the study for linear elastic problem

Data-set generation. The data-set used for training the linear PGNNIV corresponds to a **linear isotropic material** of elastic modulus $E = 10^7$ Pa and Poisson's coefficient $\nu = 0.3$. We train the network with a data-set of biaxial load-cases generated according to the load profile described in Equations (2.43) and (2.45).

Hyper-parameters. We train the network with the following hyper-parameters:

Data-set size	10000
Size of the minibatch	10000
Learning rate	0.05
$\boldsymbol{p_1}$	10^{13}
$\boldsymbol{p_2}$	10^{-9}
$\boldsymbol{p_3}$	10^{-9}
$\boldsymbol{p_4}$	10^{13}

Table 3.1: Hyper-parameters of the linear network with linear material.

Results. We calculate how accurate the prediction of the parameters E and ν is. Also, if $\boldsymbol{C}$ is the stiffness matrix of the linear material, and $\hat{\boldsymbol{C}}$ is the predicted matrix by the network containing the predicted parameters, we compute the error as:

$$\epsilon_r(\boldsymbol{C}) = \frac{(\hat{\boldsymbol{C}} - \boldsymbol{C}) : (\hat{\boldsymbol{C}} - \boldsymbol{C})}{\boldsymbol{C} : \boldsymbol{C}} \tag{3.9}$$

and we compute the relative errors of E and ν as:

$$\epsilon_r(E) = \frac{|E - \hat{E}|}{E} \tag{3.10}$$

$$\epsilon_r(\nu) = \frac{|\nu - \hat{\nu}|}{\nu} \tag{3.11}$$

Network errors			
Type of network	$\epsilon_r(E)$ (%)	$\epsilon_r(\nu)$(%)	$\epsilon_r(\boldsymbol{C})$(%)
Isotropic	0.0633	0.2622	0.076450
Orthotropic	2.244	1.481	6.855
Anisotropic	29.641	0.21	7.14

Table 3.2: Errors for the linear network

The results are shown in Table 3.2. The prediction of the elastic tensor is better for the isotropic material, followed by the orthotropic and then the anisotropic. This is the expected result since the real simulated material is isotropic. The reason of this error amplification is that the explanatory network falls into overfitting, throwing worse predictions of the stiffness parameters. The more extra parameters are included in the explanatory model, the more over-fitting we observe, while keeping the errors low enough (except, perhaps, for the Young modulus estimation in the anisotropic case). For this reason, $\epsilon_r(E)$ and $\epsilon_r(\boldsymbol{C})$ are lower than for the anisotropic case.

3.2. Infinitesimal nonlinear elasticity

Our aim is now to generalize the main ideas and methods presented for linear elasticity to nonlinear materials. The differences are in the definition of the topology of the predictive and explanatory networks, Y and H, as they must be able to recreate the non-linearities both in the relation $\boldsymbol{f} \mapsto \boldsymbol{u}$ and $\boldsymbol{\varepsilon} \mapsto \boldsymbol{\sigma}$.

3.2.1. Constitutive modeling of nonlinear materials using PGNNIV

Once we have tackled the linear material problem in the previous sections, we now aim to extend the PGNNIV concept to nonlinear materials. The software framework is Tensorflow@Python and the theoretical framework is one based on the equations presented in Section 2.2.2.

The most significant difference between the problem addressed now and the linear problem is that before we knew that the relation between external variables (displacements and forces), Y, is strictly linear, since we have defined the material as linear and the other equations related to the problem (equilibrium and compatibility) are linear differential equations. The same observation applies to the relation between internal variables (strains and stresses), H, which relation is also linear.

Consequently, now it is not possible to define a stiffness matrix $\boldsymbol{C}$ relating stress and strain variables, and we need to define a nonlinear relation. In the neural network language, that corresponds to an arbitrary network H able to capture the nonlinear relation. This is possible thanks to the universal approximation theorem [41]. For solving the problem, we add internal layers (also known as hidden layers) to the neural network model with a particular activation function so we can provide the network with the learning capability and complexity that the new nonlinear constitutive law would require. In conclusion, we should be able to learn a constitutive law that is nonlinear.

Nonlinear network topology for the nonlinear elastic materials (softening and hardening materials). According to the model of the board described in section 2.1.4, this object is discretized in 80 elements (8×10 cm^2). Based on this assumption, the network topology is discussed considering the displacement at each node and the strains and stresses at each element. At the same time, external loads are considered as pressure on each element of the boundary. As explained in Chapter 2, we can distinguish between two sub-networks: the one used for capturing the dependence between applied

force on each node (or pressure on each element) and the resultant displacement at the node, and the one used for unraveling the material behavior or constitutive law. We now describe the topology of both sub-networks. The link between this two sub-networks is the calculation of the strains out of the predicted displacements (output from the Y sub-network).

1. Y **network topology.** A biaxial compressive quadratic load applied now on both right and top faces requires from a complex neural network where the top and right load components are merged in a sum. The result of this sum is input to another neural network which provides the resultant displacement. In this case, the first two sub-neural networks have 10 neurons and 1 layer (right face and top face) and the sub-neural network after the sum contains 3 layers of 10 neurons each, as shown in Figure 3.2. A more detailed description of the dimensions of the weights and biases matrices and the mathematical modeling of the network is provided in the following lines:

   The structure described above and represented in Figure 3.2 is shown in the next equations:

   Input data (loads): $\boldsymbol{f^t}$, $\boldsymbol{f^r}$

   Output data (displacements): $\boldsymbol{u}$

$$
\begin{aligned}
\boldsymbol{x}_{11} &= \phi(\boldsymbol{f^t} \cdot \boldsymbol{W}_1 + \boldsymbol{b}_1) \\
\boldsymbol{x}_{12} &= \phi(\boldsymbol{f^r} \cdot \boldsymbol{W}_2 + \boldsymbol{b}_2) \\
\boldsymbol{x}_2 &= \phi(\boldsymbol{x}_{11} + \boldsymbol{x}_{12}) \\
\boldsymbol{x}_3 &= \phi(\boldsymbol{x}_2 \cdot \boldsymbol{W}_b + \boldsymbol{b}_b) \\
\boldsymbol{x}_4 &= \phi(\boldsymbol{x}_3 \cdot \boldsymbol{W}_c + \boldsymbol{b}_c) \\
\boldsymbol{x}_5 &= \phi(\boldsymbol{x}_4 \cdot \boldsymbol{W}_d + \boldsymbol{b}_d) \\
\boldsymbol{u} &= \phi(\boldsymbol{x}_5 \cdot \mathbf{W}_e + \mathbf{b}_e)
\end{aligned}
\tag{3.12}
$$

   where ϕ is the hyperbolic tangent activation function.

   Figure 3.2 shows a graphical representation of 3.12:

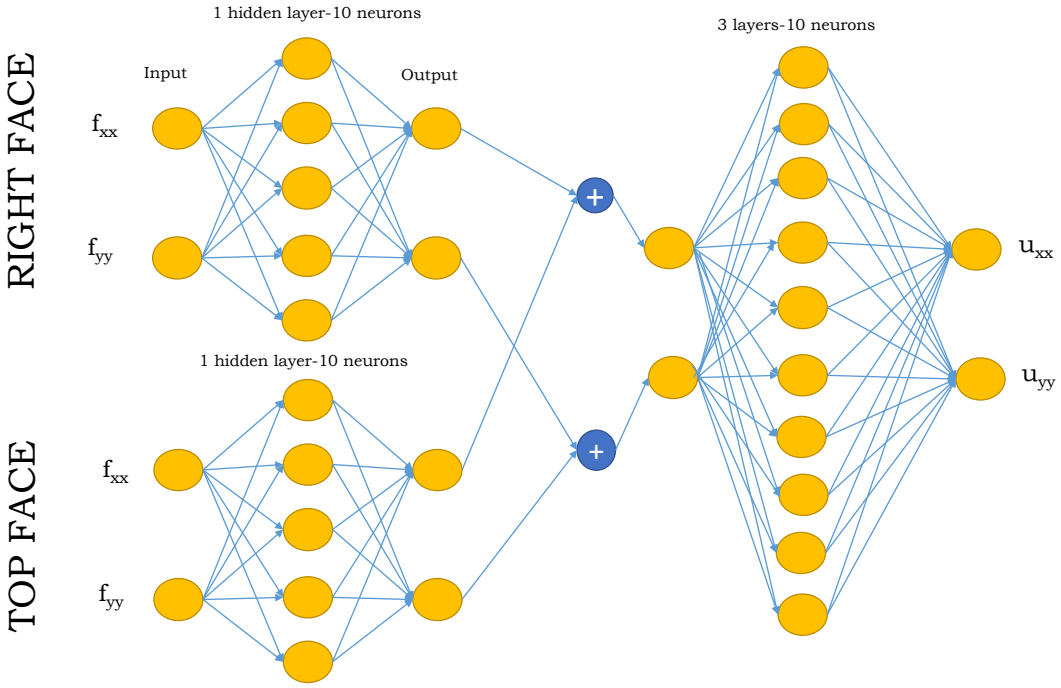


Figure 3.2: Biaxial test sub-network $force - displacement$ topology

2. H **network topology.** This sub-network contains 6 layers of 50 neurons each and correlates the two internal variables of our problem, that is, the strains and the stresses. For that reason we work with images of pixels representing the board, where each pixel corresponds to an element to which corresponds the value of the aforementioned internal variable, and the weights and biases build the constitutive law.

   Each three-dimensional strain pixel, contains the three components of the plane strain deformation vector, $(\varepsilon_{xx}, \varepsilon_{xy}, \varepsilon_{yy})$. When inputting this vector into the neural network, we obtain the three components of the stress pixel $(\sigma_{xx}, \sigma_{xy}, \sigma_{yy})$. This network works according to the following algorithm: each ε element is the input of a pixel-wise *deconvolution* neural network that outputs the plane stresses $(\sigma_{xx}, \sigma_{xy}, \sigma_{yy})$. The network itself (containing the same weights and biases for each pixel) scans the whole image moving along its both dimensions. A simplified representation of the process is shown in Figure 3.3, where only one of the 6 layers and 5 neurons instead of 50 are represented.

   First, the algorithm expands each elemental strain value (three components) to a number of neurons and layers and then it squeezes these layers to end up with

the three elemental components of the stress value. It is important to remark that this algorithm links exclusively pixels with the same spacial coordinates i, j. Therefore, it is specifically designed for local (homogeneous) materials, as the one from the case of study. Further details about the mathematical structure of the network and the *deconvolution* algorithm can be found in Appendix D.

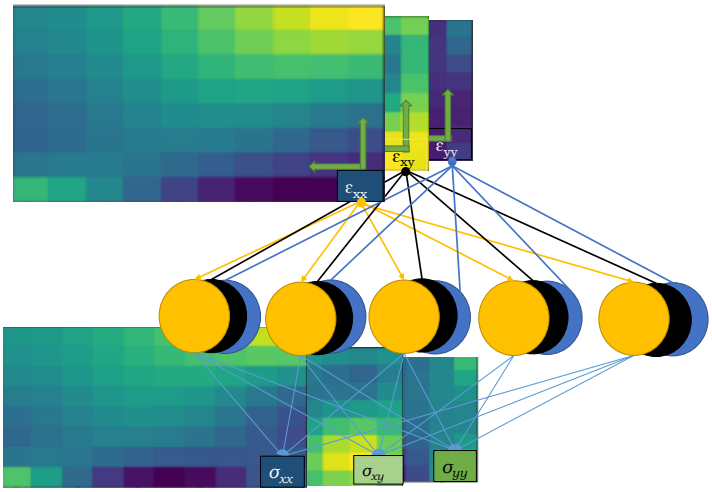


Figure 3.3: Deconvolutional network representation.

The use of Y and H together results in our PGNNIV, which from now on is named as nonlinear PGNNIV topology. The topology-related parameters of of this network are summarize in Table 3.3:

Number of layers of the Y network	5
Number of neurons per layer of the Y network	10
Number of layers of the H network	6
Number of neurons per layer of the H network	50

Table 3.3: Topology-related parameters of of the nonlinear PGNNIV topology.

3.2.2. Creation of the nonlinear elastic materials

For the creation of the materials, we use Abaqus CAE/6.14-2 to define two different nonlinear materials, one showing softening and another showing hardening properties. We disable the option for finite strains `NLGEOM` (since the materials are nonlinear elastic under the infinitesimal strain theory framework) that Abaqus provides and we define a nonlinear elastic material. Once the mathematical model is created, we proceed to simulate a uniaxial (both right and top load independently) compression test to build up the traction curves of the materials (axes shown in Figure 2.2), as presented in Figure 3.4. This curves represent the constitutive law of the nonlinear elastic materials. The constitutive law for the softening material is:

$$\sigma = 18.69\varepsilon^{0.45} \tag{3.13}$$

whereas the one associated with the hardening material is:

$$\sigma = 1828.5\varepsilon^{3} \tag{3.14}$$

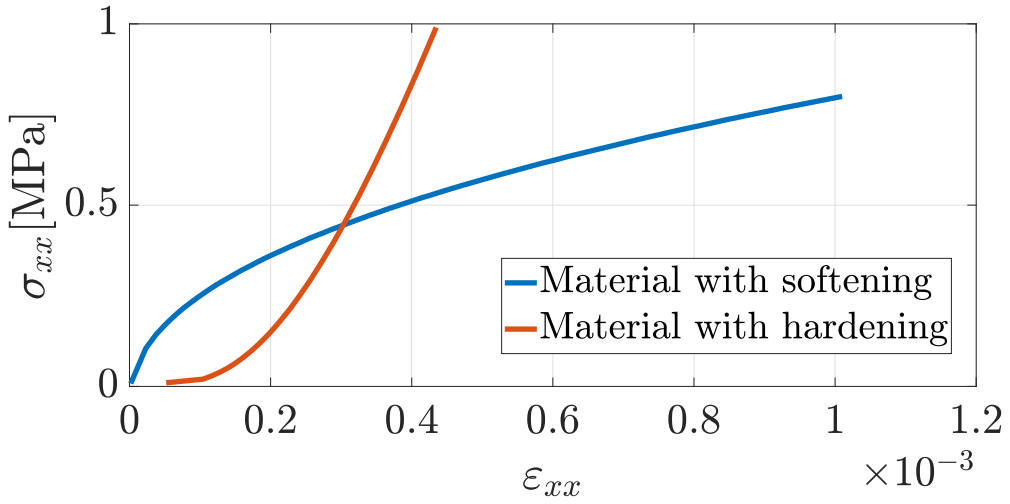


Figure 3.4: Curve for uniaxial traction test for the two nonlinear elastoplastic materials.

H network testing using FEM based data. Now the aim is to test the network topology when using the values of strains and stresses obtained from Abaqus generated dataset. This test will ensure that the numerical noise due to FEM approximation has a low impact on the computations. We suppose the simulation to be realistic since we use Abaus CAE, which uses finite element approximation for solving the nonlinear problem with both materials.

Figures 3.5 shows the train and tested σ-ε curves, where each point represents the pair (ε, σ) of a given element, identified with a pixel, when the material is brought under uniaxial test conditions (load on the right face of the board), where the blue points represents the predicted values for each pixel (element) and the red points corresponds to the simulated ones. The dispersion in the cloud of points, when compared to Figure 3.5 is due to both Poisson effect and to the discretization error associated with the FEM simulation.

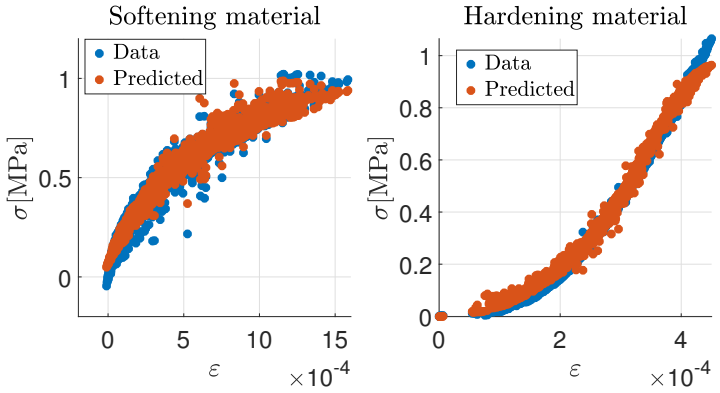


Figure 3.5: Stress prediction for both materials

The Mean Squared Error (MSE) and relative error (MSE_r) for both materials are shown in Table 3.4, and are calculated as follows:

$$\text{MSE} = \frac{1}{N}\sum_{i=1}^{N} ||\bar{\sigma}^i - \mathsf{H}(\varepsilon^i)||^2 \tag{3.15}$$

where $\mathsf{Y}(\varepsilon^i)$ is the prediction on each layer.

$$\text{MSE}_\text{r} = \sqrt{\frac{\sum_{i=1}^{N} ||\bar{\sigma}^i - \mathsf{H}(\varepsilon^i; \mathbf{W})||^2}{\sum_{i=1}^{N} ||\bar{\sigma}^i||^2}} \tag{3.16}$$

These errors have satisfactory values (very low values for both softening and hardening) and lead us to think that this explanatory neural network works when we train the load-displacement network with data from displacements and external forces.

Stress prediction errors		
Type of material	MSE (MPa^2)	MSE_r (%)
Softening	0.0022	3.9
Hardening	0.0024	3.7

Table 3.4: Prediction errors for the softening and hardening materials with the H sub-network.

3.2.3. Results for the nonlinear problem

In this section we present the results obtained after training the nonlinear PGNNIV with data-sets corresponding to softening and hardening materials.

For evaluating the results, we use RMSE (Mean Squared Errors) and RE (Relative Errors). We use the notation RMSE(X) to refer to the RMSE associated with a given variable X. If X is a scalar variable, such as u_x or ε_{xx}, we will use as norm the absolute value, whereas if $\boldsymbol{X}$ is a vectorial or tensorial variable, such as $\boldsymbol{u}$ or $\boldsymbol{\varepsilon}$, we will use as norm the standard Frobenius norm. For the relative error we follow the same rule. These errors are defined as:

The Root Mean Squared Error of the variable X reads:

$$\mathrm{RMSE}(X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(||\hat{I}^i(X) - I^i(X)||)^2} \tag{3.17}$$

where N is the size of the batch of test images (simulations/samples) and $\hat{I}^i$ and I^i are the predicted and real image fields respectively associated with the considered variable X.

The Relative Error of the variable X reads:

$$\mathrm{RE}(X) = \sqrt{\frac{\sum_{i=1}^{N}||\hat{I}^i(X) - I^i(X)||^2}{\sum_{i=1}^{N}||I^i(X)||^2}} \tag{3.18}$$

The relative errors per pixel (element or node, depending whether we refer to strains/stresses or displacements) are analyzed for the predicted displacements and defined as follows. The mean relative error per pixel for a given variable X is:

$$\mu_{kl}(X) = \frac{1}{N}\sum_{i=1}^{N}\frac{|\hat{I}^i_{kl}(X) - I^i_{kl}(X)|}{|I^i_{kl}(X)|} \tag{3.19}$$

Finally, the standard deviation is defined as:

$$\sigma_{kl}(X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}\left(\frac{|\hat{I}^i_{kl}(X) - I^i_{kl}(X)|}{|I^i_{kl}(X)|} - \mu_{kl}(X)\right)^2} \tag{3.20}$$

SOFTENING MATERIAL

In this section we present the most relevant results obtained when training and testing the nonlinear PGNNIV with the described softening material. A comparison between the linear network topology and the nonlinear topology is also performed for all variables. Displacements fields are in cm while stress fields are MPa.

Training process: This table summarizes the hyper-parameters used for the training of the network with the softening material data-set:

Dataset size	10000
Size of the minibatch	100
Learning rate	0.00005
$\boldsymbol{p_1}$	64000
$\boldsymbol{p_2}$	0.001
$\boldsymbol{p_3}$	0.001
$\boldsymbol{p_4}$	64000

Table 3.5: Hyper-parameters of the training process for the softening material.

In order to illustrate the network performance, we show, in Figures 3.6, 3.7 and 3.8, the nonlinear PGNNIV predictions of all variables (that is, displacements, strains and stresses) and the two networks approaches (linear and nonlinear PGNNIV) for one particular load-case belonging to the test data-set (that is, that has not been used during the training process).

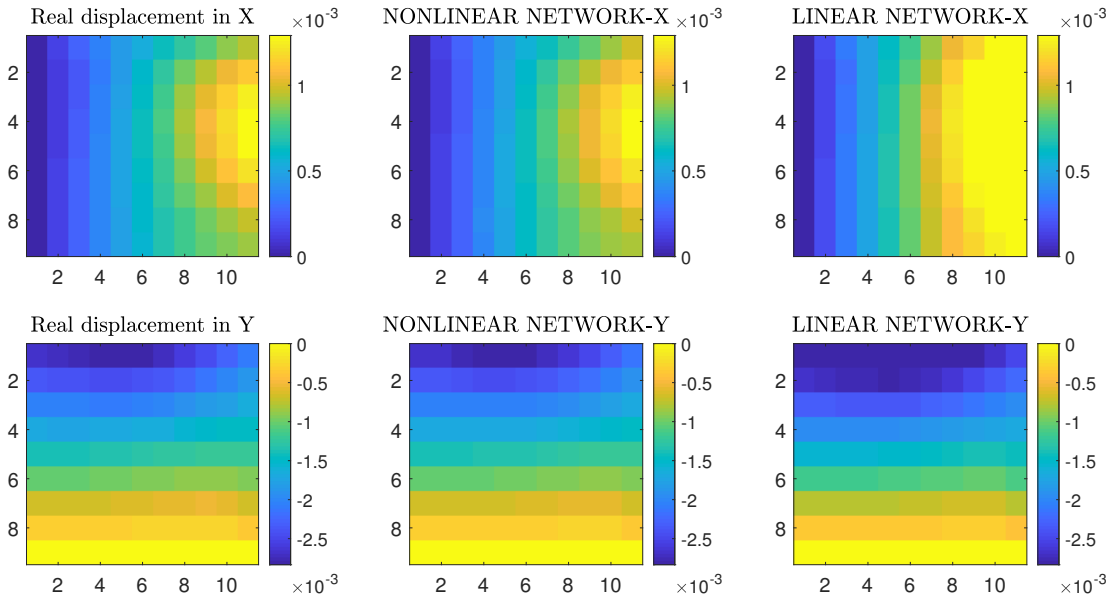


Figure 3.6: Displacements components prediction for softening material (cm).

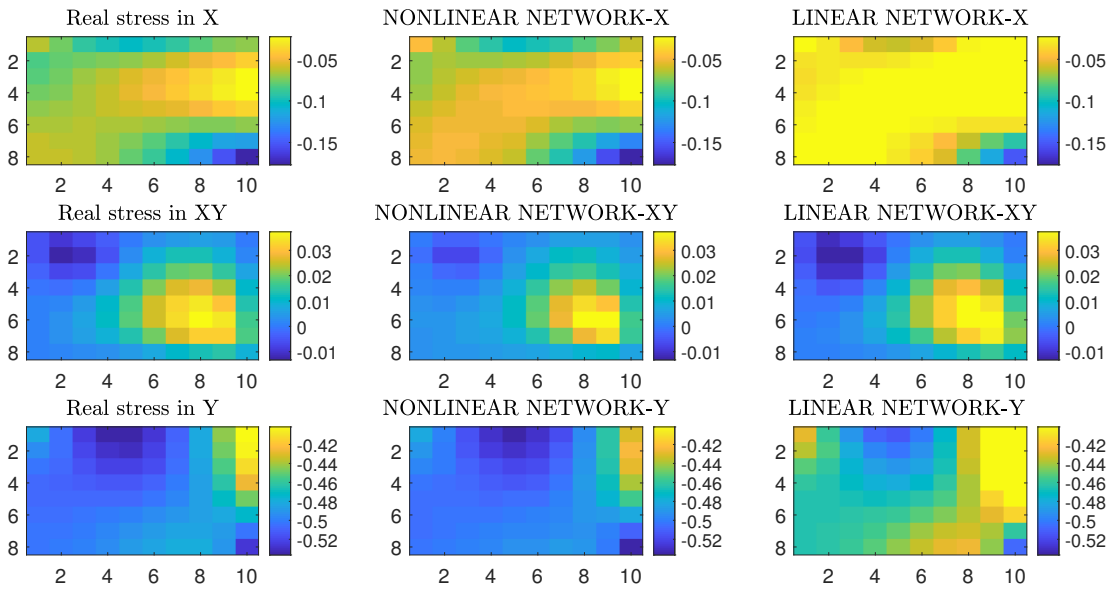


Figure 3.7: Stress components prediction for softening material (MPa).

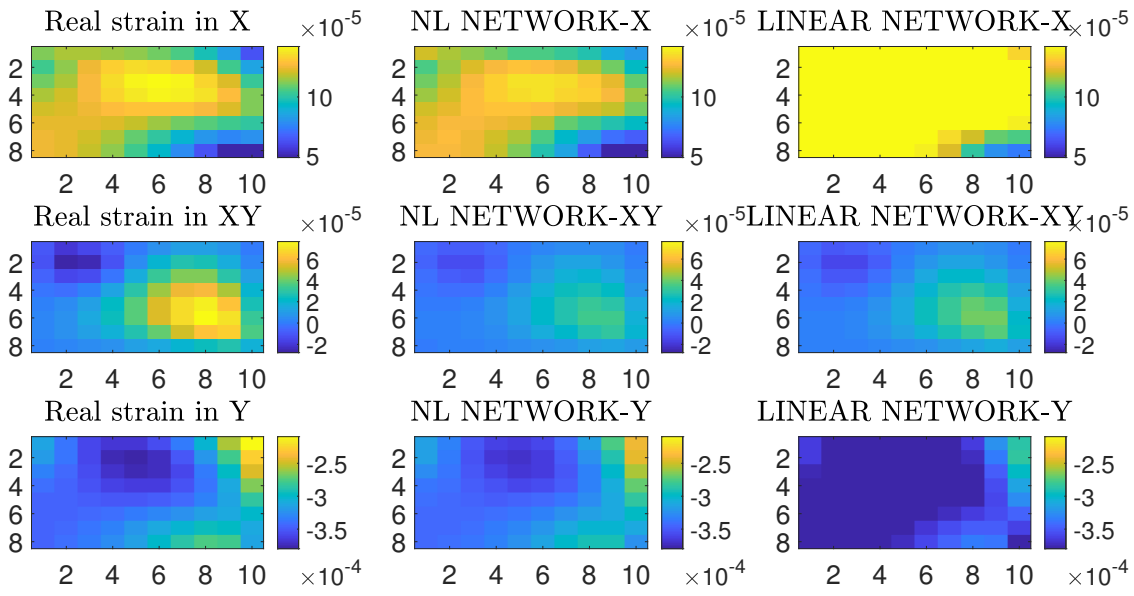


Figure 3.8: Strain components prediction for softening material.

Prediction errors						
	RMSE($\boldsymbol{u}$) (cm)	RMSE($\boldsymbol{\sigma}$) (MPa)	RMSE($\boldsymbol{\varepsilon}$)	RE($\boldsymbol{u}$)	RE($\boldsymbol{\sigma}$)	RE($\boldsymbol{\varepsilon}$)
Linear	8.57×10^{-5}	0.636	6.75×10^{-4}	0.1816	0.11	0.19
Nonlinear	9×10^{-6}	0.16	2×10^{-4}	0.019	0.029	0.058

Table 3.6: Errors for the softening material external and internal variables.

It is clear that this error has to be evaluated for the whole test data-set to check if the network gives accurate results in general. This is done next by computing the statistical indicators given in Equations (3.19) and (3.20) associated with displacements, strains and stresses. Besides, Table 3.6 shows the value of the errors given in Equation (3.17) and (3.18), illustrating that the predictions for the softening material have been improved one order of magnitude with respect to the linear network. In the Figures 3.9, 3.10 and 3.11 we present the mean relative error (Equation (3.19)) of all variables involved in the problem.

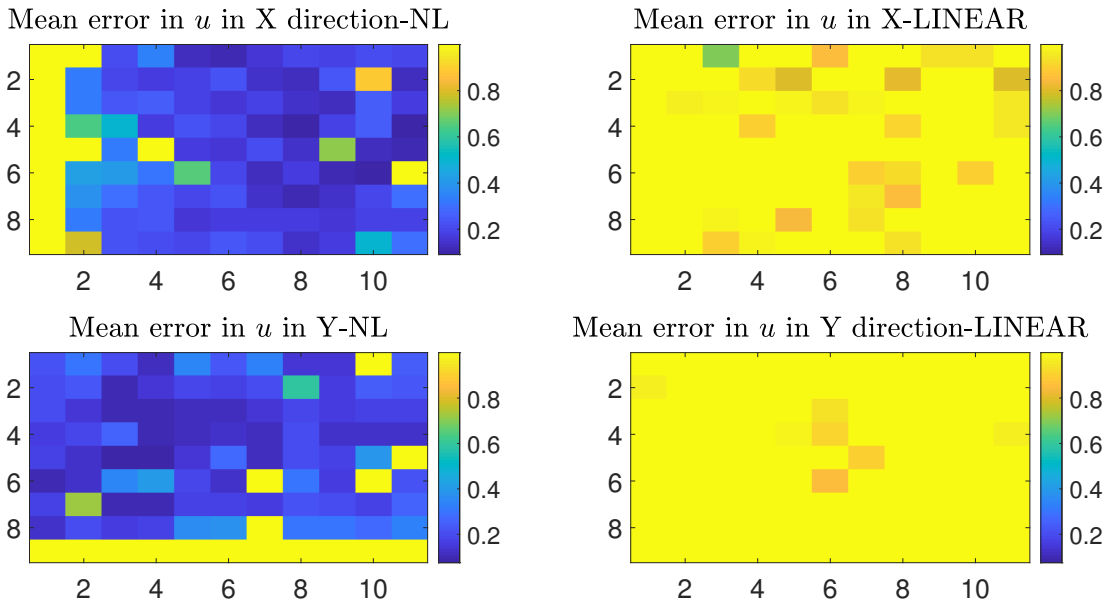


Figure 3.9: Mean relative error per pixel of the nonlinear PGNNIV for softening material (u).

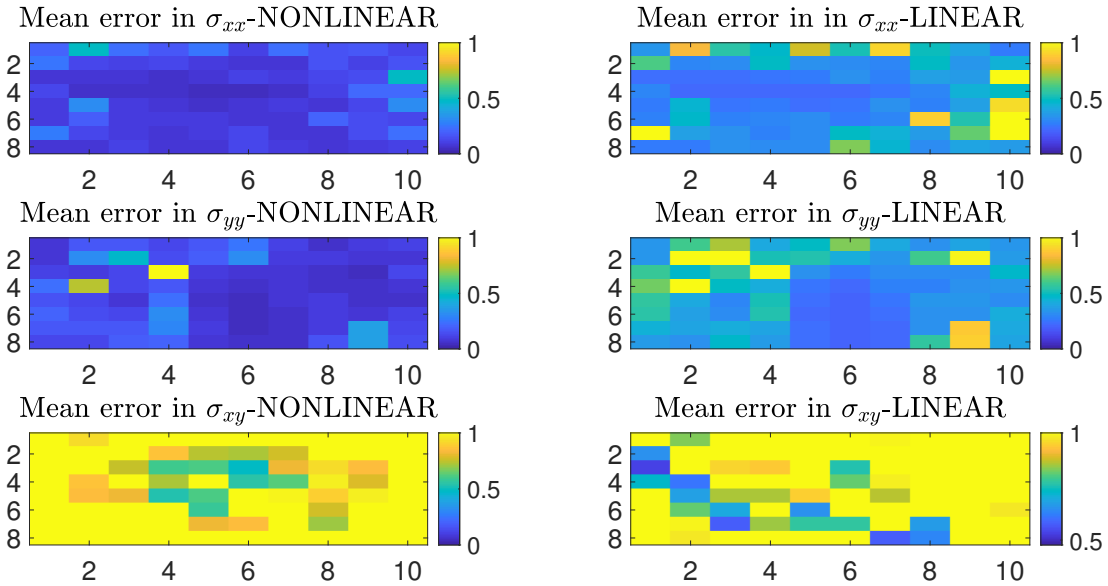


Figure 3.10: Mean relative error per pixel for softening material (σ).

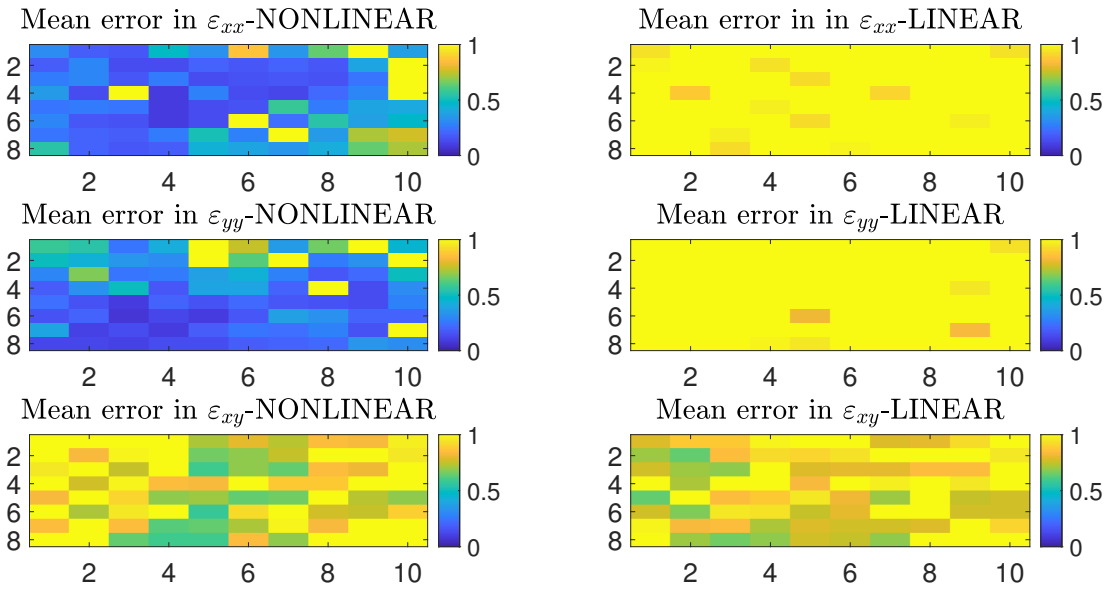


Figure 3.11: Mean relative error per pixel for softening material (ε).

The main conclusions extracted from Figures 3.9, 3.10 and 3.11 are:

– In Figure 3.9, the first column of pixels has values tending to infinity for all images in the X direction, as well as for the last row in the Y direction. This is expected since we divide by a very small quantity when calculating the relative errors per pixel (see Equation (3.19)) corresponding to the encastre boundary condition.

– Errors for all variables are small for the nonlinear PGNNIV prediction, except for σ_{xy} and ε_{xy}. That means the nonlinear PGNNIV produces exact results, but since no relevant shear load is applied in any sample of the training data-set, these components are not predicted accurately.

– The nonlinear PGNNIV prediction for the softening material outperforms the linear network. This fact will be discussed later more quantitatively.

HARDENING MATERIAL

In this section we present the most relevant results obtained when training and testing the nonlinear PGNNIV with the described hardening material. A comparison between the linear network topology and the nonlinear topology is also performed for all variables. Displacements fields are in cm while stress fields are MPa.

Training process: This table summarizes the hyper-parameters used for the training of the network with the hardening material data-set:

Dataset size	1000
Size of the minibatch	100
Learning rate	0.000005
$\boldsymbol{p_1}$	64000
$\boldsymbol{p_2}$	0.001
$\boldsymbol{p_3}$	0.001
$\boldsymbol{p_4}$	64000

Table 3.7: Hyper-parameters of the training process for the hardening material.

Figures 3.12, 3.13 and 3.14 show the prediction of the nonlinear and linear PGNNIV.

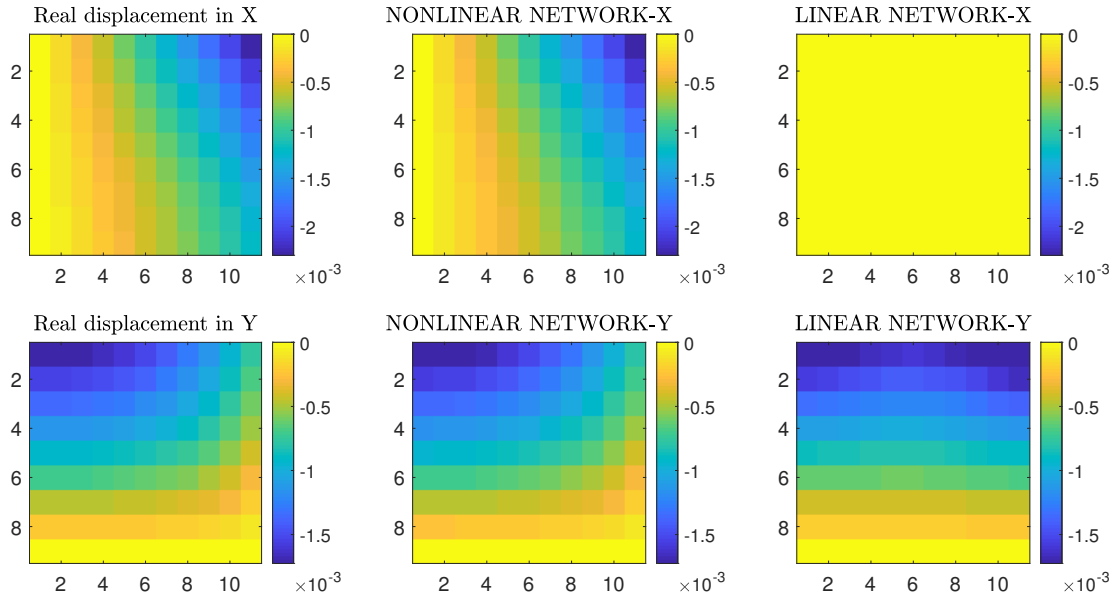


Figure 3.12: Displacements components prediction of the nonlinear PGNNIV for hardening material (cm).

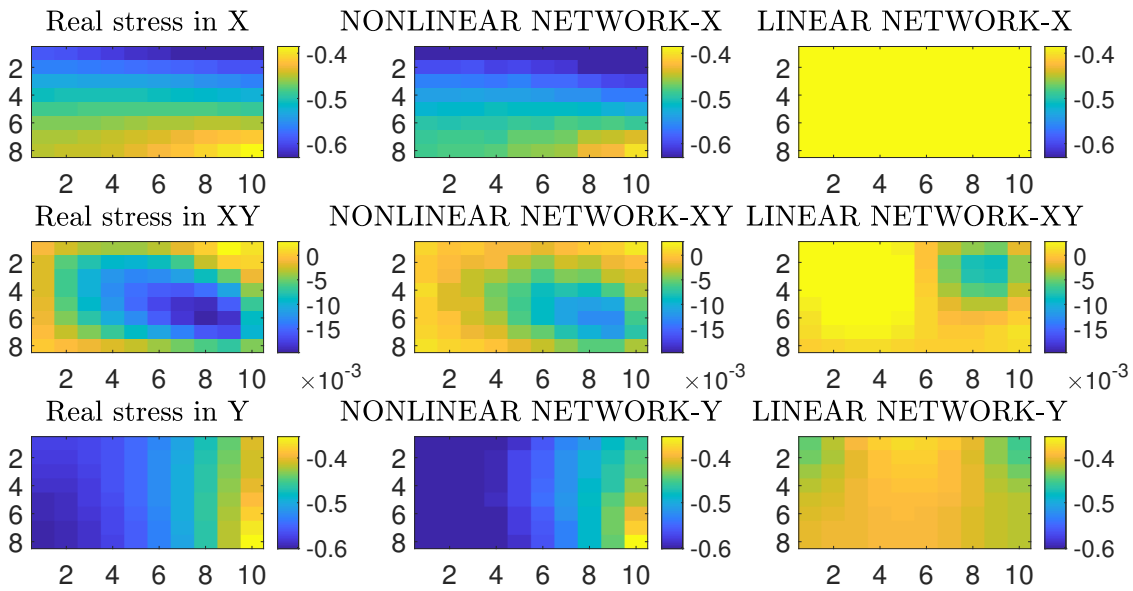


Figure 3.13: Stress components prediction of the nonlinear PGNNIV for hardening material.

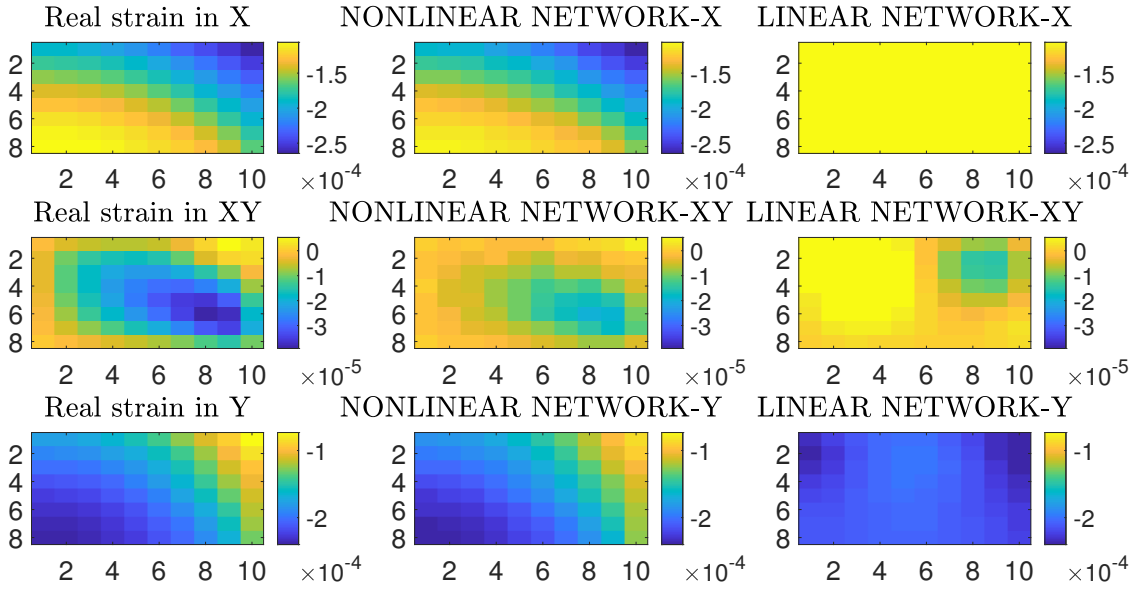


Figure 3.14: Strain components prediction of the nonlinear PGNNIV for hardening material (MPa).

Prediction errors						
	RMSE($\boldsymbol{u}$) (cm)	RMSE($\boldsymbol{\sigma}$) (MPa)	RMSE($\boldsymbol{\varepsilon}$)	RE($\boldsymbol{u}$)	RE($\boldsymbol{\sigma}$)	RE($\boldsymbol{\varepsilon}$)
Linear	1.37×10^{-4}	0.52	3.7×10^{-4}	0.133	0.09	0.15
Nonlinear	3.31×10^{-5}	0.4	1.9×10^{-4}	0.032	0.07	0.076

Table 3.8: Errors for the hardening material external and internal variables.

The predicted errors for hardening material presented in Table 3.8 are, for all variables, smaller than the ones yielded by the linear network. However, the PGNNIV demonstrates a better performance on the softening material (see Table 3.6) than in the hardening material.

In Figures 3.15, 3.16 and 3.17 we present the mean relative errors for all variables.

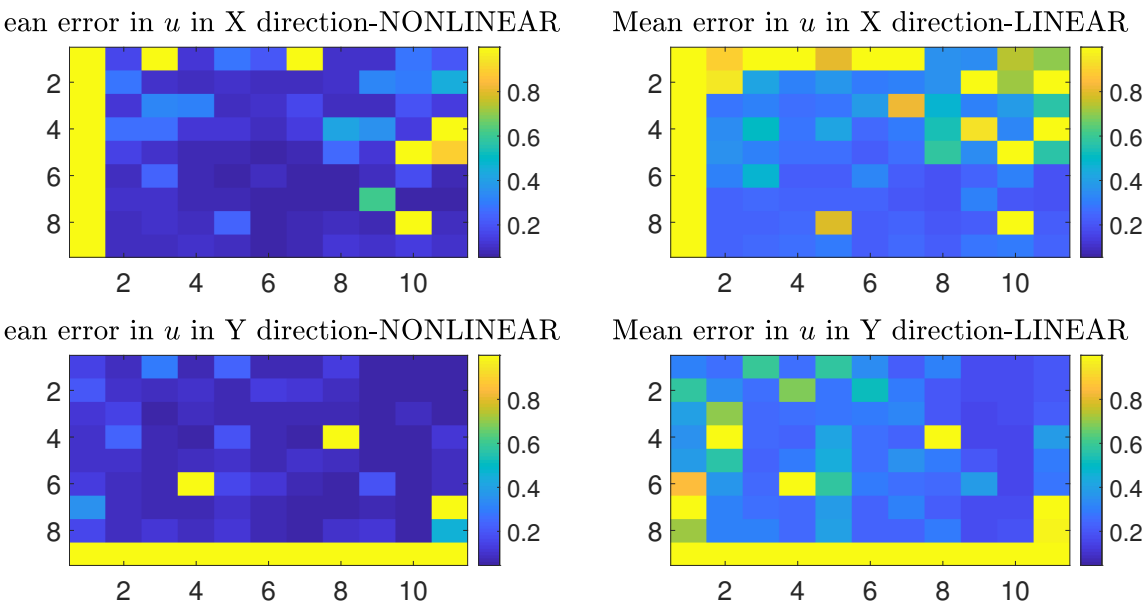


Figure 3.15: Mean relative error per pixel of the nonlinear PGNNIV for hardening material (u).

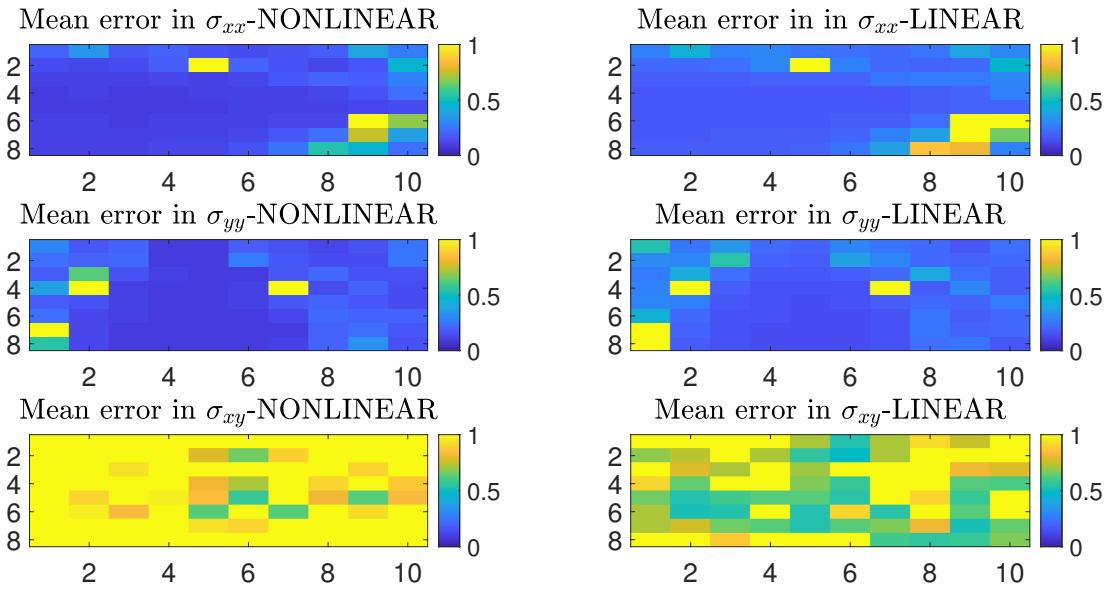


Figure 3.16: Mean relative error per pixel of the nonlinear PGNNIV for hardening material (σ).

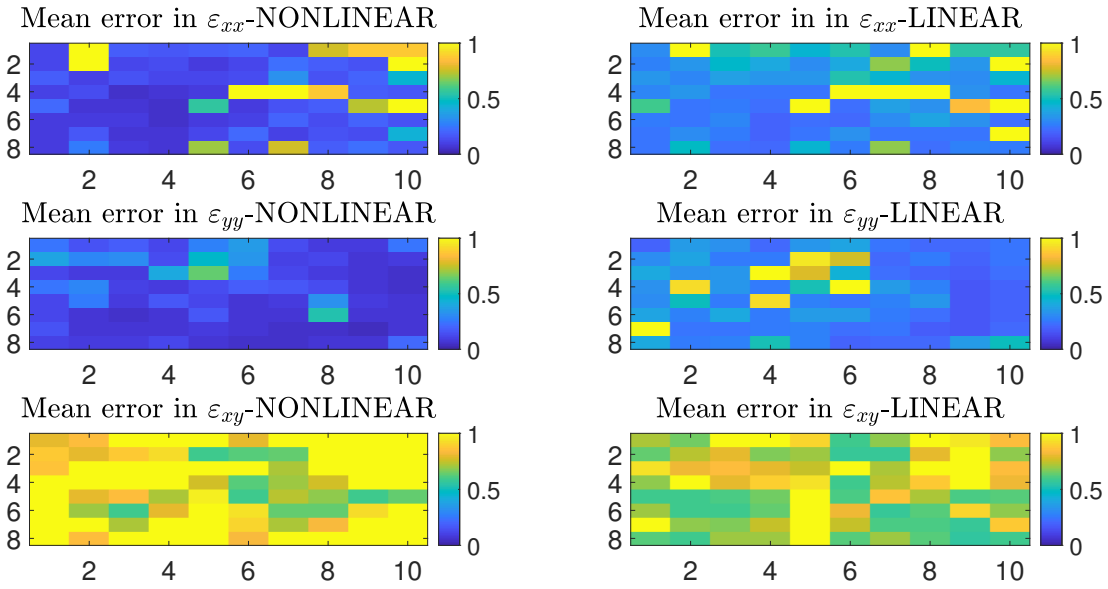


Figure 3.17: Mean relative error per pixel of the nonlinear PGNNIV for hardening material (ε).

The main conclusions extracted from Figures 3.15, 3.16 and 3.17 are:

- Errors for all variables are small for the nonlinear PGNNIV prediction, except for σ_{xy} and ε_{xy}. That means the nonlinear PGNNIV produces exact results, but since no relevant shear load is applied at any sample of the training data-set, these components are not predicted accurately.

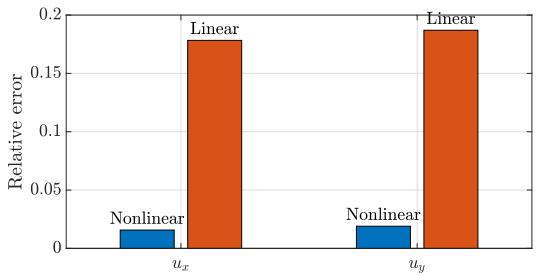
- The nonlinear PGNNIV prediction for the softening material outperforms the linear network, but not as much as it did with the softening material. The reason why this occurs is twofold: On one hand, this might be caused by the fact that the data-set size is smaller (1000 samples for the hardening material instead of the 10000 samples that we used for the softening). On the other hand, the hyperbolic tangent activation function is prone to saturation for values that are close to 1, which is the case of the hardening material we use (see Figure 3.4).

3.3. Quantitative comparison between linear and nonlinear PGNNIV

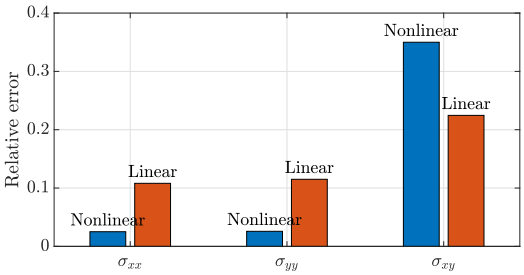
3.3.1. Predictive capacity

We now summarize the results obtained in Section3.2.3 by comparing the RE error for all the involved variables (displacements, strains and stresses in all the directions). We can recall the definition of RE of a variable X from Equation (3.18).

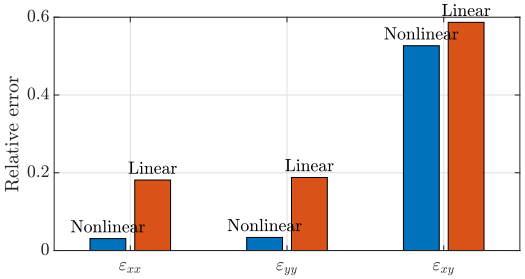
Figure 3.18 shows the comparison between the prediction of the linear and nonlinear PGNNIV for the softening material.



(a) RE(u) for the softening material.



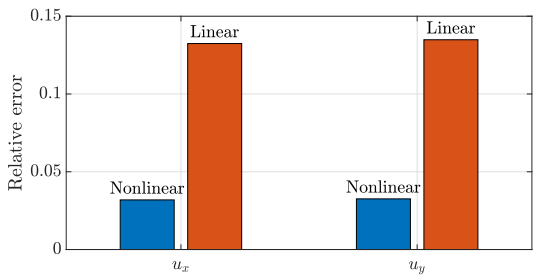
(b) RE(σ) for the softening material.



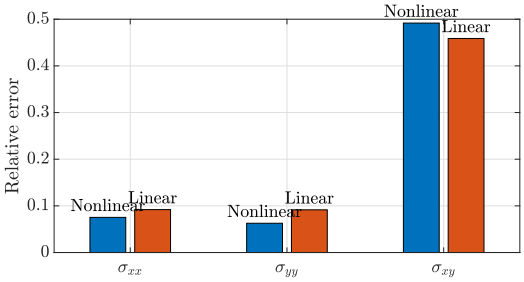
(c) RE(ε) for the softening material.

Figure 3.18: Relative global errors for the softening material.

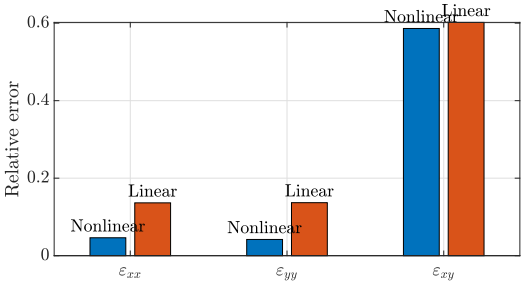
Figure 3.19 shows the comparison between the prediction of the linear and nonlinear PGNNIV for the hardening material.



(a) RE(u) for the hardening material.



(b) RE(σ) for the hardening material.



(c) RE(ε) for the hardening material.

Figure 3.19: Relative global errors for the hardening material.

From Figures 3.18 and 3.19, we can conclude that:

– The developed nonlinear network yields smaller errors for the displacements and for all variables associated with the normal directions in comparison with the linear network.

– For the shear fields, that is ε_{xy} and σ_{xy}, neither the nonlinear nor the linear PGNNIV predict values accurately. This is due to the fact that the PGNNIV is not trained with a load that can produce high enough shear strains and stresses, so that the network can learn and accurately predict these values. Therefore, this is not a problem of the model H, but of a lack of variability of the training data-set.

3.3.2. Explanatory capacity

The explanatory capacity of the network is assessed by its ability to learn the material constitutive law. In order to evaluate the explanatory capacity of the network, we perform a virtual uniaxial test (as the one illustrated in Figure 3.20) to the input of the explanatory network H and we compare the stress predictions with the ones obtained for the actual material. The results are shown in Figure 3.21 for the softening material and in Figure 3.22 for the hardening material. We input constant stress profiles increasing their value from 0 to 0.8 MPa (since the data-set also ranges through these values) to generate the traction curve of the softening material, and ranging from 0 to 1 MPa for the hardening material.

The PGNNIV is able to reproduce the nonlinear behavior of the softening material.

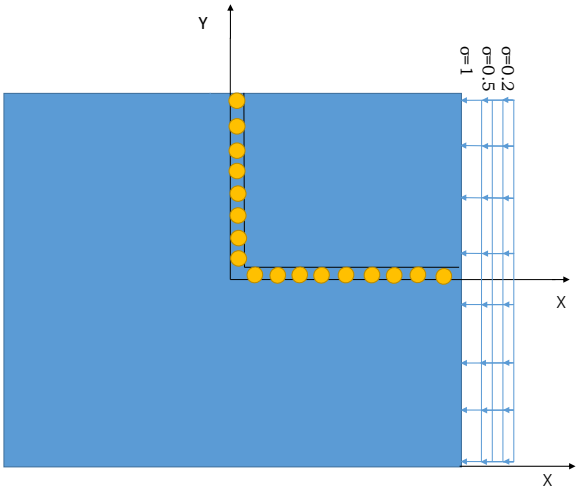


Figure 3.20: Uniaxial compression test.

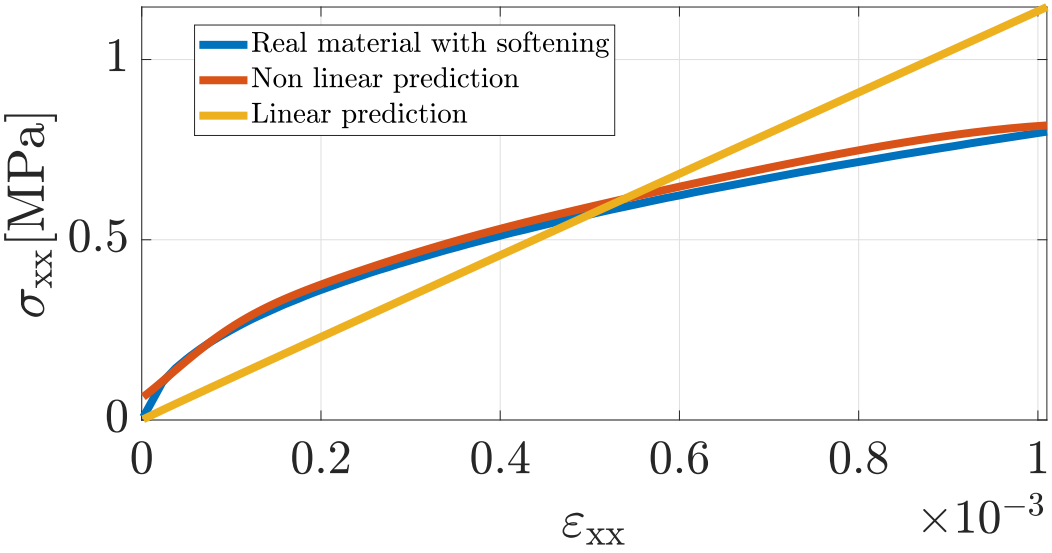


Figure 3.21: Stress prediction for softening material.

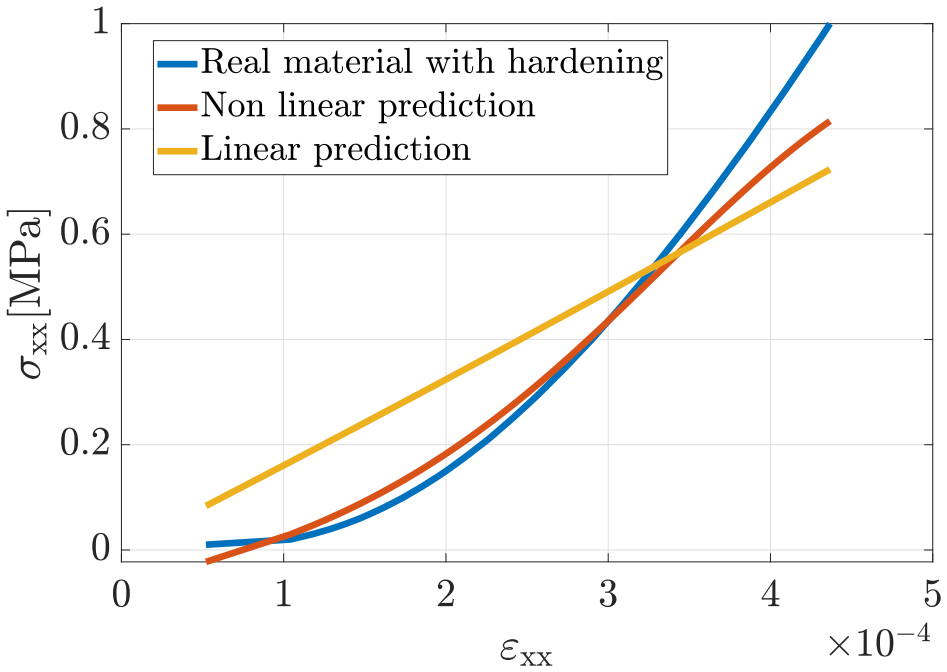
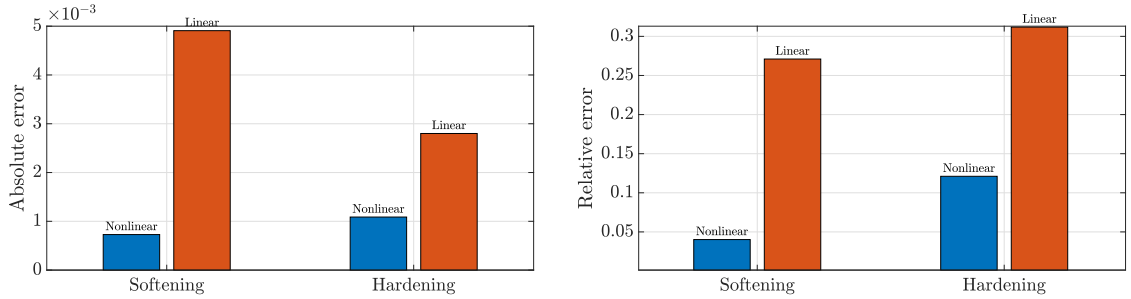


Figure 3.22: Stress prediction for hardening material.

The reproduction of the material behavior for the hardening material is also good, although it is worse than for the softening material (see Figure 3.21), specially for high values of the stress due to saturation of the hyperbolic tangent. In any case, the nonlinear PGNNIV outperforms the linear PGNNIV for both materials.



(a) Absolute error between curves (MJ/m^3). (b) Relative error between curves.

Figure 3.23: Explanatory error.

In order to summarize the information contained in these plots we calculate the explanatory error between the real material curve and the predicted one as follows:

$$\epsilon(W) = \sqrt{\int_{\varepsilon_{min}}^{\varepsilon_{max}} (\hat{\sigma} - \sigma)^2 \, d\varepsilon} \tag{3.21}$$

where $\hat{\sigma}$ is the observed stress value and σ the predicted one. The relative error is:

$$\epsilon_r(W) = \sqrt{\frac{\int_{\varepsilon_{min}}^{\varepsilon_{max}} (\hat{\sigma} - \sigma)^2 \, d\varepsilon}{\int_{\varepsilon_{min}}^{\varepsilon_{max}} (\sigma)^2 \, d\varepsilon}} \tag{3.22}$$

where ε_{max} is the maximal strain experimented by the board and ε_{min} is approximately 0.

In Figure 3.23a we observe that the explanatory error is always smaller for the nonlinear prediction. Even though, it is clear that this effect is more accentuated for the softening material.

Chapter 4

Towards real material modelling: hyperelastic and test-based materials

4.1. Contextualization

We intend to extend the developed network configuration to the hyperelastic family. In order to do so, we create a Neohookean material and a hyperelastic test-based polynomial material, we simulate a biaxial test on them and we train the nonlinear PGNNIV with the simulated data-set.

4.2. Neohookean material

4.2.1. Neo-Hookean constitutive relation

We create a Neo-Hookean material model. For the Neo-Hookean material considered, the strain energy potential [42] is given by:

$$\Psi = C_{10}(\overline{I_1} - 3) + \frac{1}{D_1}(J-1)^2 \tag{4.1}$$

where Ψ is the strain energy per unit of reference volume; $C_{10} = 1.5$ and $D_1 = 1$; $\overline{I_1}$ is the first deviatoric strain invariant defined as:

$$\overline{I_1} = \overline{\lambda_1}^2 + \overline{\lambda_2}^2 + \overline{\lambda_3}^2 \tag{4.2}$$

where the deviatoric stretches $\overline{\lambda_i} = J^{-1/3}\lambda_i$ and J is the total volume ratio. The parameters used to define this material provide it with a linear behavior within the infinitesimal strain theory.

4.2.2. Data-set generation

The data-set used to train the network was produced by a biaxial-uniaxial parabolic simulation test, that is, biaxial and uniaxia load profiles acting on the right and top free surfaces. As a result of the 10000 simulations performed, in Figures 4.1 and 4.2 we illustrate the character of the constitutive law for the pairs of state tensors $\boldsymbol{\varepsilon} - \boldsymbol{\sigma}$:

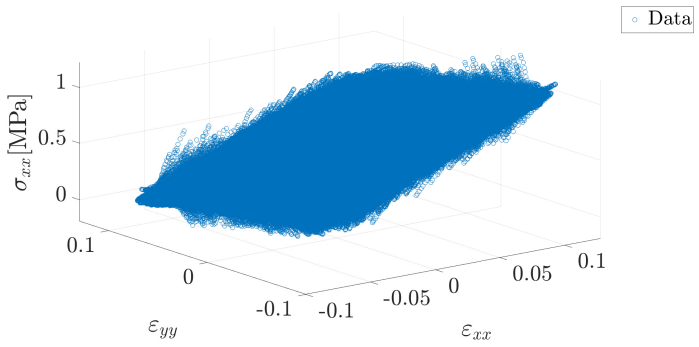


Figure 4.1: Neohookean material values for the σ_{xx} stresses as a function of the strains.

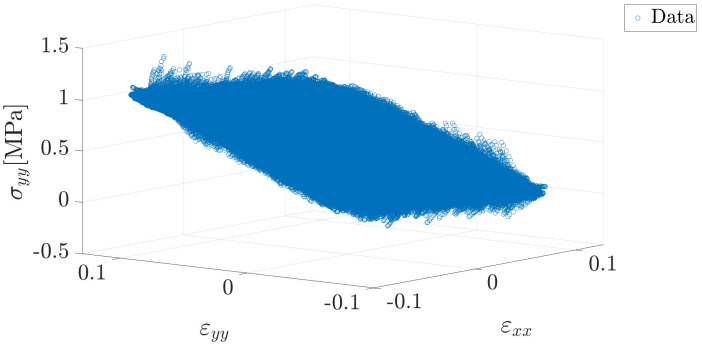


Figure 4.2: Neohookean material values for the σ_{yy} stresses as a function of the strains.

Figure 4.1 and 4.2 show the high linearity of the Neohookean material, for both σ_{xx} and σ_{yy}. The different values of the triplets $(\varepsilon_{yy}, \varepsilon_{xx}, \sigma_{xx})$ and $(\varepsilon_{yy}, \varepsilon_{xx}, \sigma_{yy})$ are therefore in a plane.

4.2.3. Results

Training process: This table summarizes the hyper-parameters used for the training process:

Data-set size	10000
Size of the minibatch	100
Learning rate	0.0005
$\boldsymbol{p_1}$	64000
$\boldsymbol{p_2}$	0.001
$\boldsymbol{p_3}$	0.001
$\boldsymbol{p_4}$	64000

Table 4.1: Hyper-parameters of the training process for the Neohookean material.

Prediction results. An example of the prediction for all variables is given in Figures 4.3, 4.4 and 4.5.

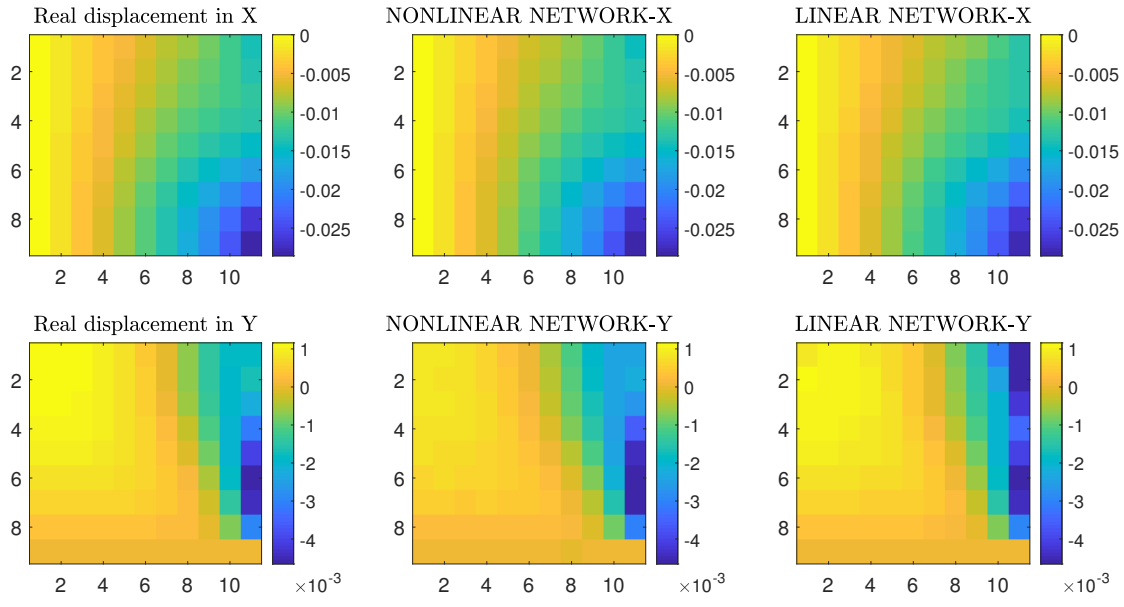


Figure 4.3: Displacement components prediction for the Neohookean material (cm).

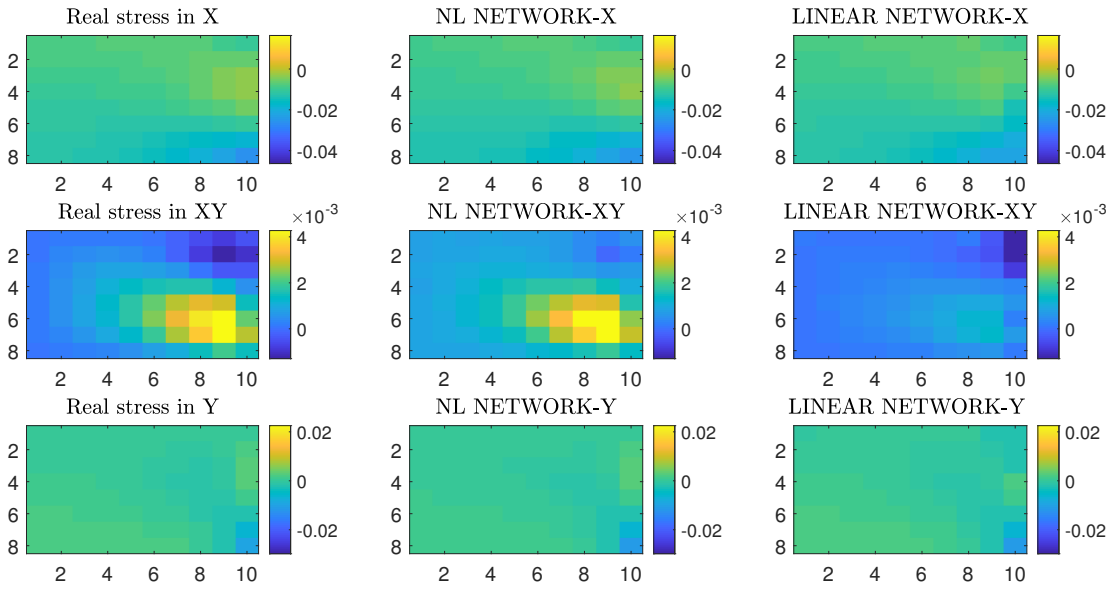


Figure 4.4: Stress components prediction for the Neohookean material (MPa).

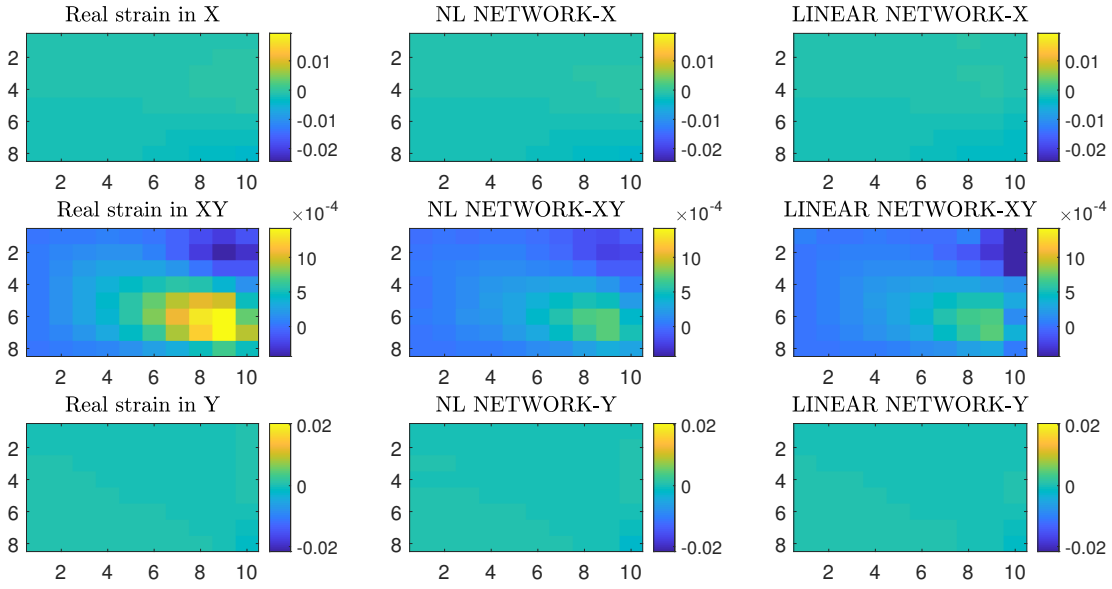


Figure 4.5: Strain components prediction for the Neohookean material.

Prediction errors						
	RMSE($\boldsymbol{u}$) (cm)	RMSE($\boldsymbol{\sigma}$) (MPa)	RMSE($\boldsymbol{\varepsilon}$)	RE($\boldsymbol{u}$)	RE($\boldsymbol{\sigma}$)	RE($\boldsymbol{\varepsilon}$)
Nonlinear	4.56×10^{-5}	0.01	0.003	0.0026	0.013	0.023
Linear	4.56×10^{-4}	0.019	0.004	0.0075	0.02	0.029

Table 4.2: Errors for the Neohookean material external and internal variables.

As we can observe from Table 4.2, prediction errors from both linear and nonlinear networks are both accurate. This is due to the highly linear nature of the data, as seen in Figure 4.1 and 4.2.

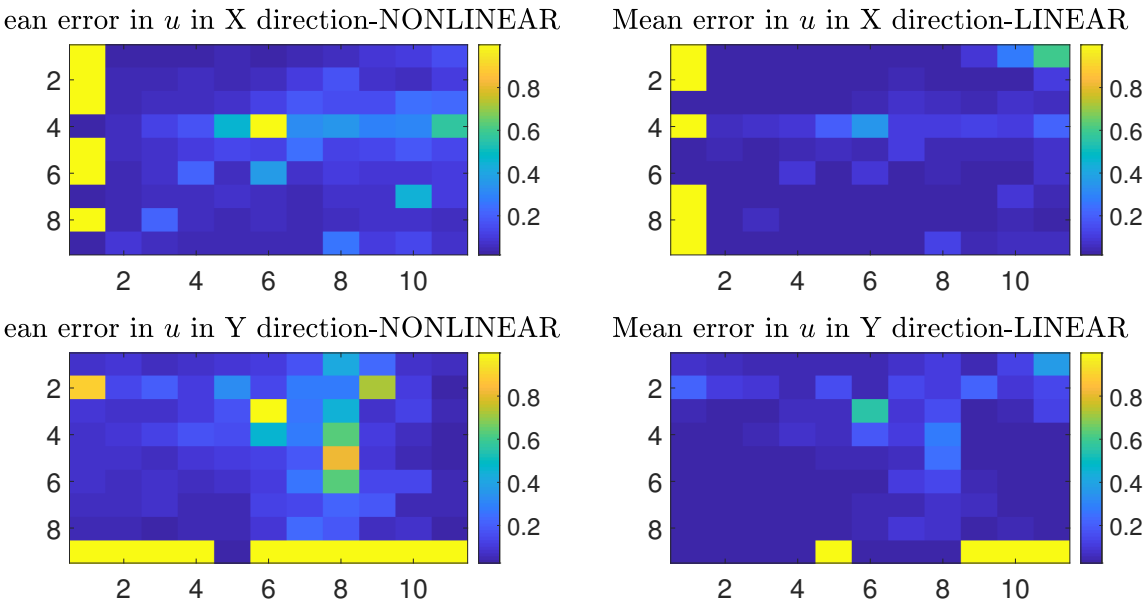


Figure 4.6: Mean relative error per pixel for the Neohookean material (u).

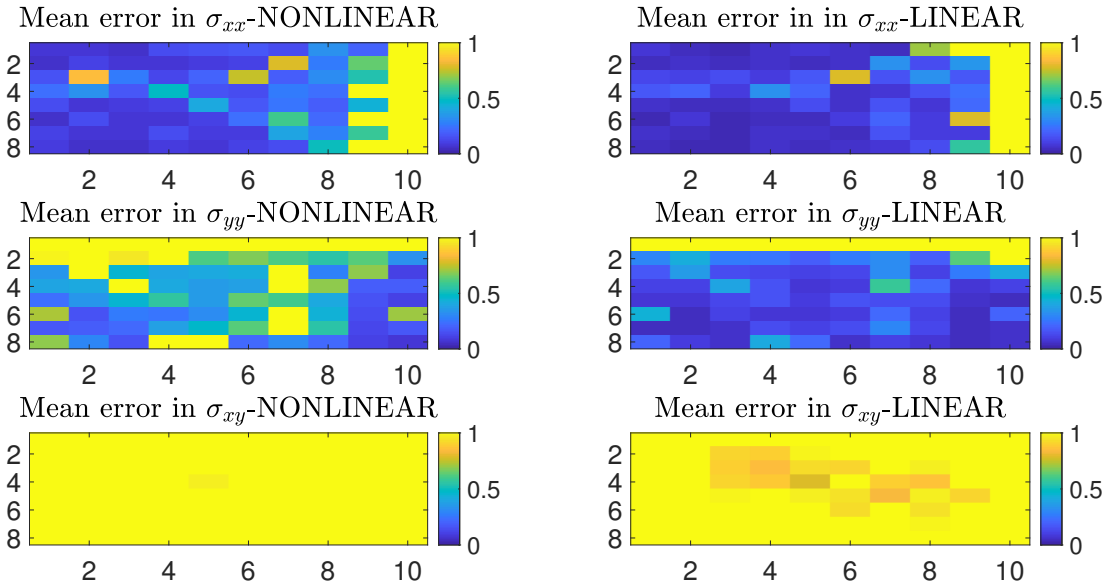


Figure 4.7: Mean relative error per pixel for the Neohookean material (σ).

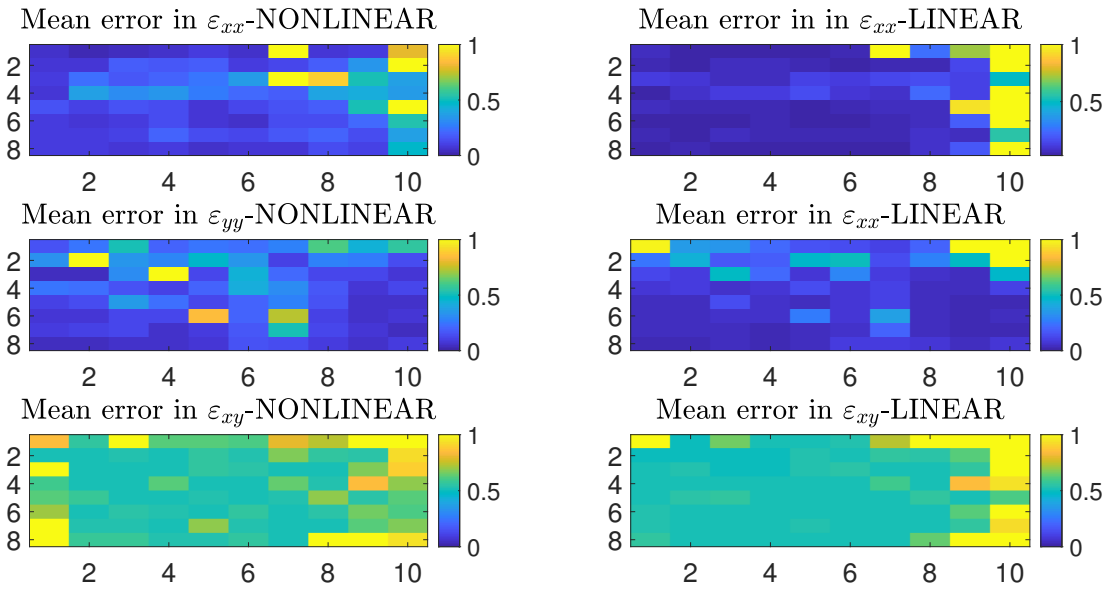


Figure 4.8: Mean relative error per pixel for the Neohookean material (ε).

In Figures 4.6, 4.7 and 4.8, the mean relative error predictions of the nonlinear network are in most cases similar to the linear one. This is due to the fact that the Neohookean material is chosen close to linear in the strains' and stresses' magnitude range of study. We can also observe that the prediction of σ_{xy} is not good. This is again due to the lack of shear loads on the boundary.

State equation prediction.

Figures 4.9, 4.10 and 4.11 show the predicted value of the stress for each pixel with respect to the simulated one with the nonlinear PGNNIV.

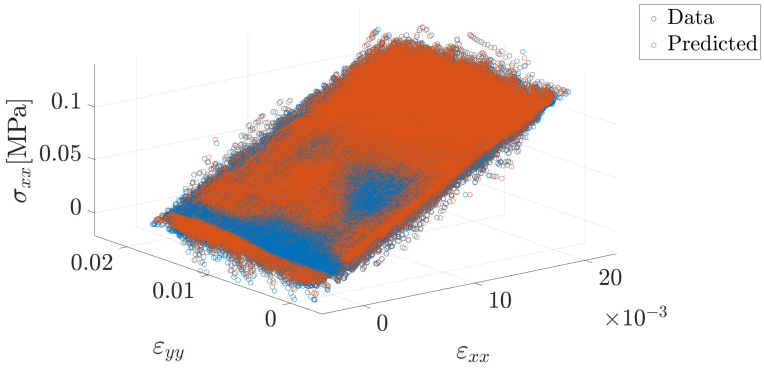


Figure 4.9: Stress prediction σ_{xx} for the Neohookean material with the nonlinear PGNNIV.

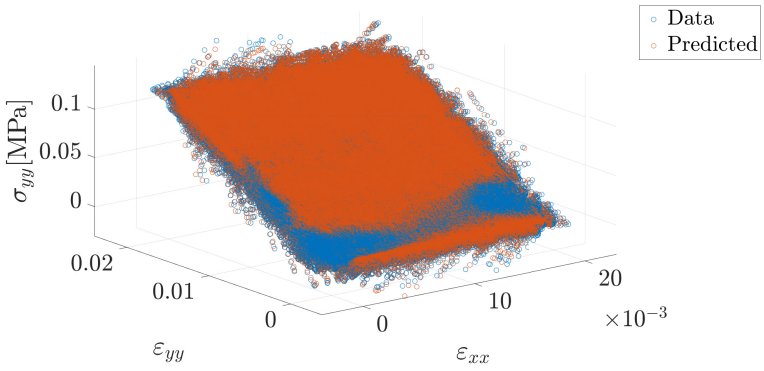


Figure 4.10: Stress prediction σ_{yy} for the Neohookean material with the nonlinear PGNNIV.

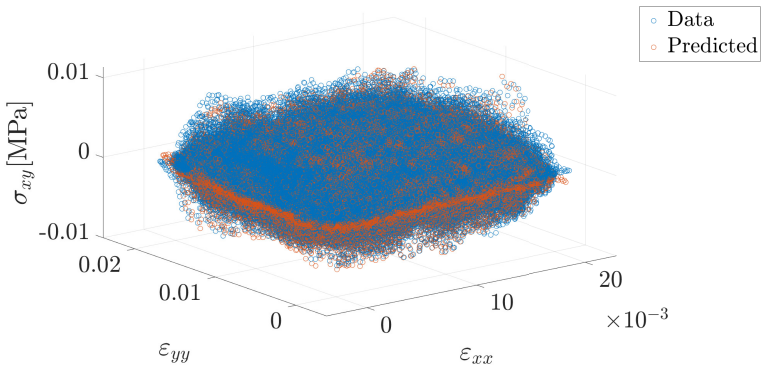


Figure 4.11: Stress prediction σ_{xy} for the Neohookean material with the nonlinear PGNNIV.

We observe that the constitutive law $\boldsymbol{\sigma}$-$\boldsymbol{\varepsilon}$ is well predicted. This conclusion is enforced by Figure 4.12, where we can see a 3D plot of the three components of the $\boldsymbol{\varepsilon}$ tensor, where the components of the $\boldsymbol{\sigma}$ tensor are mapped by assigning a color grade to each point of the $(\varepsilon_{xx}, \varepsilon_{yy}, \varepsilon_{xy})$ space.

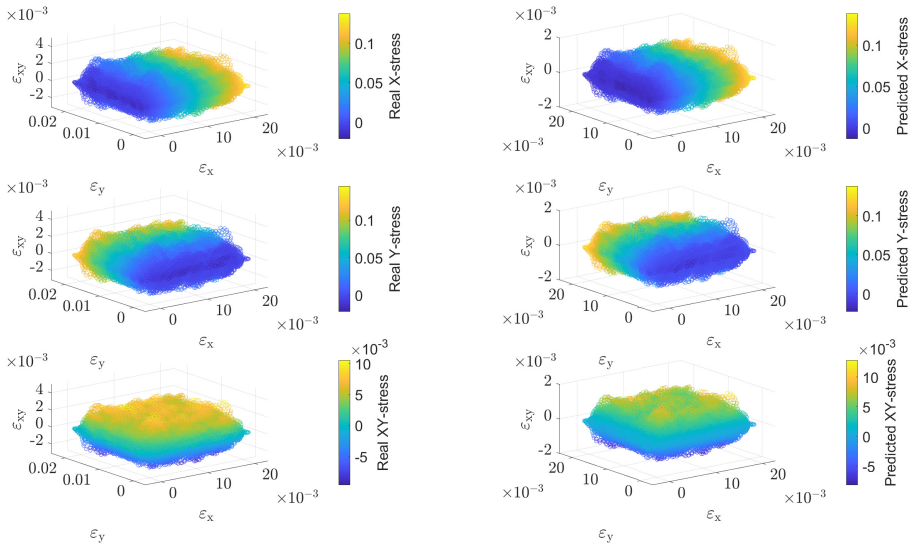


Figure 4.12: Strain prediction and stress prediction for the Neohookean material with the nonlinear PGNNIV.

Not only we can observe that the clouds of points are extremely similar when comparing left (real components of the tensor) and right (predicted components of the tensor), but also the $\boldsymbol{\sigma}$ prediction is accurate, as the similar color-maps show.

Figures 4.13, 4.14 and 4.15 show the prediction of the the linear network.

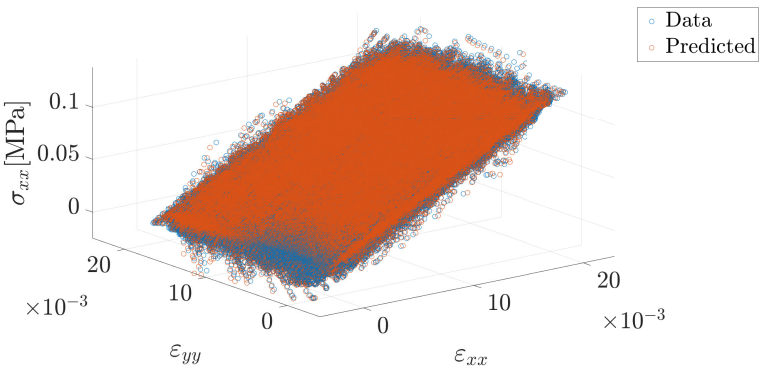


Figure 4.13: Stress prediction σ_{xx} for the Neohookean material with the linear PGNNIV.

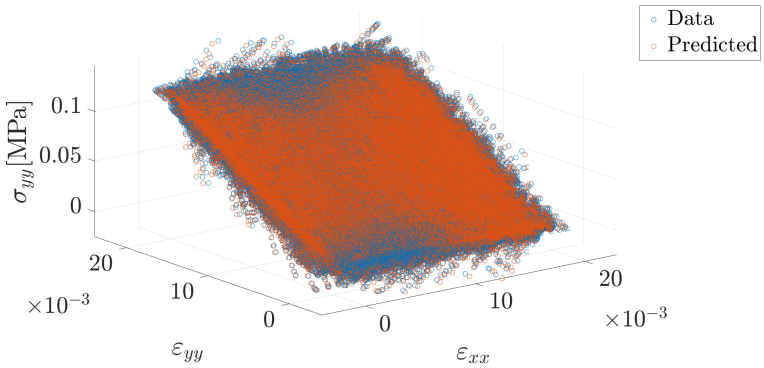


Figure 4.14: Stress prediction σ_{yy} for the Neohookean material with the linear PGNNIV.

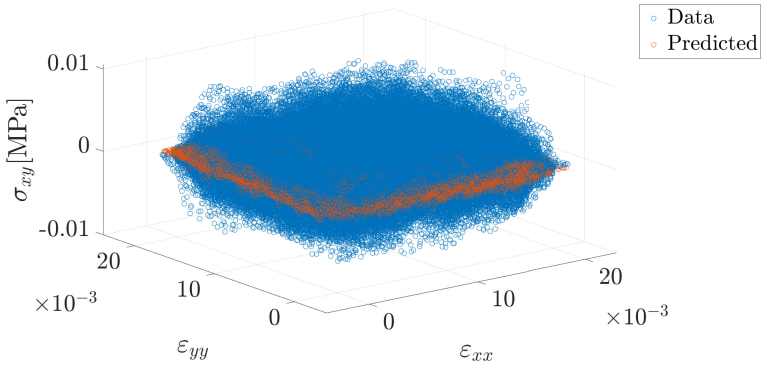


Figure 4.15: Stress prediction σ_{xy} for the Neohookean material with the linear PGNNIV.

In Figures 4.13, 4.14 and 4.15 we can observe that the linear network prediction of the Neohookean contitutive law is accurate.

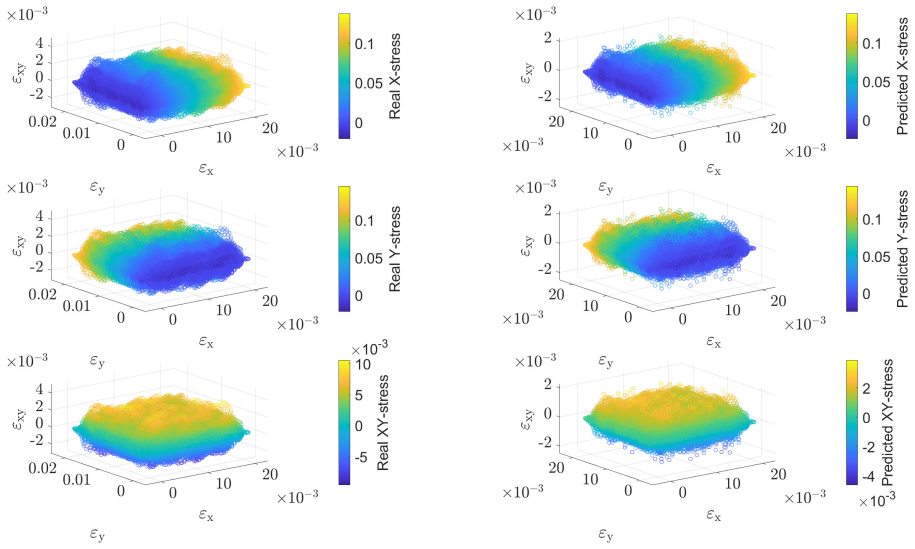


Figure 4.16: Strain prediction and stress prediction for the Neohookean material with the linear PGNNIV.

Again, this is enforced by Figure 4.16, that shows that the prediction of both internal variables is accurate. Therefore, the linear network produces accurate results for the Neohookean material.

It is important to note that for this case of analysis, the discussion about the shear response learning is very clear: as we can see in Figures 4.9, 4.10, 4.114.13, 4.14 and 4.15, the shear stresses are many orders of magnitude lower than the normal stresses, so the variability of the learning data-set does not allow the PGNNIV (neither they were linear nor nonlinear) to learn the response of the material to the shear stimuli.

4.3. Test-based polynomial material

4.3.1. Data

The test-based polynomial material is defined by means of test data, which can be found in section B. The uniaxial test data was fitted very accurately by a 2^{nd} order polynomial material, as shown in Figure 4.17.

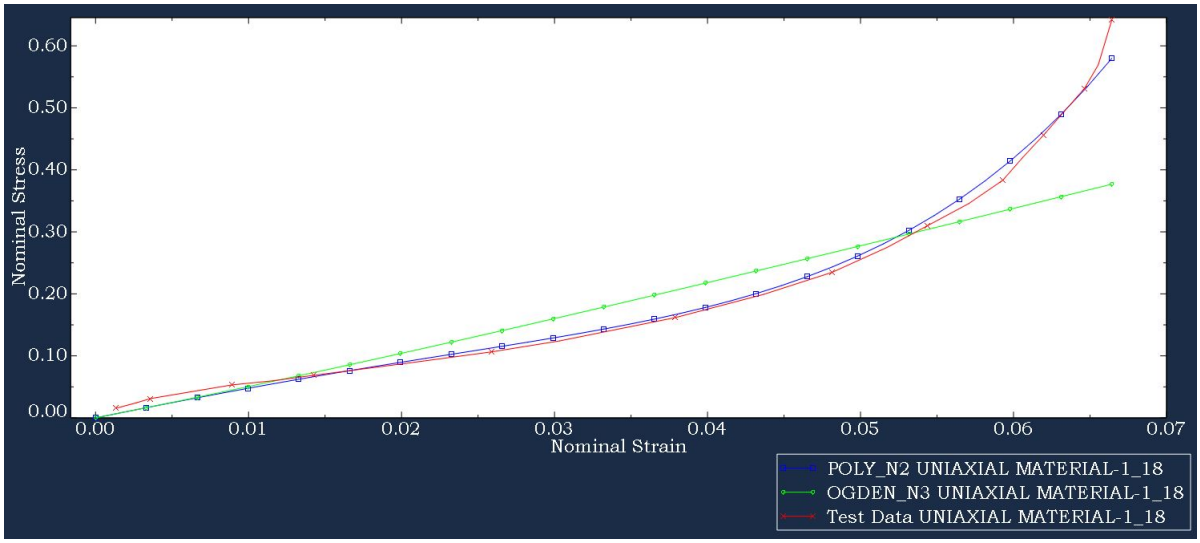


Figure 4.17: Uniaxial test fitting.

This fitting is performed by means of the Abaqus Material Evalation tool [43]. Abaqus CAE provides an option that allows the user to observe the constitutive law of different hyperelastic material models and, based on test data, choose a suitable material formulation, that is, the model and its parameters. Since Abaqus considers the strain energy density function as polynomial, we will call this test-based material as test-based polynomial material.

The data-set used to train the network was produced by a pure biaxial hydrostatic simulation test, that is, constant and identical load profiles acting on the right and top free surfaces. As a result of the 1000 simulations performed, we obtain the following constitutive law:

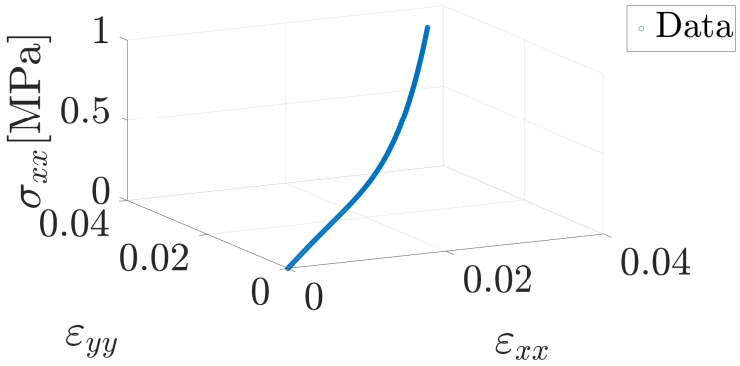


Figure 4.18: Uniaxial test data (σ_{xx}).

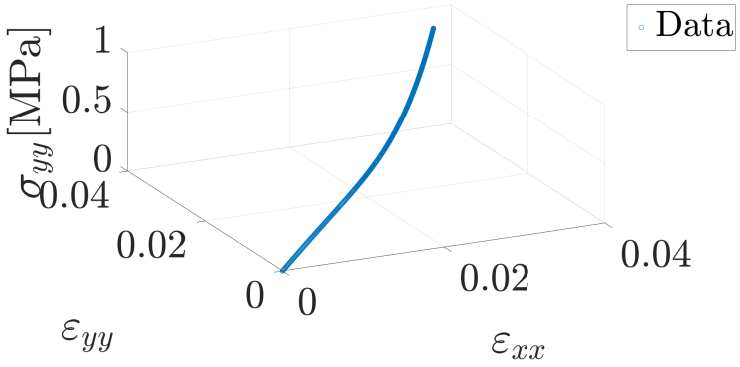


Figure 4.19: Uniaxial test data (σ_{yy}).

It can be observed that the simulated points are placed along the ε_{xx}-ε_{yy} diagonal, since no distinction is made between ε_{xx} and ε_{yy} to define the stress (the test is hydrostratic).

4.3.2. Results

We train and test the developed nonlinear PGNNIV the described data-set corresponding to a highly nonlinear test-based polynomial material under the umbrella of the infinitesimal strain theory.

Training process: This table summarizes the hyper-parameters used for the training process:

Data-set size	1000
Size of the minibatch	100
Learning rate	0.0005
$\boldsymbol{p_1}$	64000
$\boldsymbol{p_2}$	10^{-8}
$\boldsymbol{p_3}$	10^{-8}
$\boldsymbol{p_4}$	64000

Table 4.3: Hyper-parameters of the training process for the test-based polynomial material.

As we mentioned before, the simulation test done with the test-based polynomial material is performed by applying a hydrostatic, constant and biaxial load profile. An example of the prediction for all variables is given in Figure 4.20, Figure 4.21 and Figure 4.22. As it can be seen, the solution for the strains and stresses are uniform fields, due to the hydrostatic nature of the load.

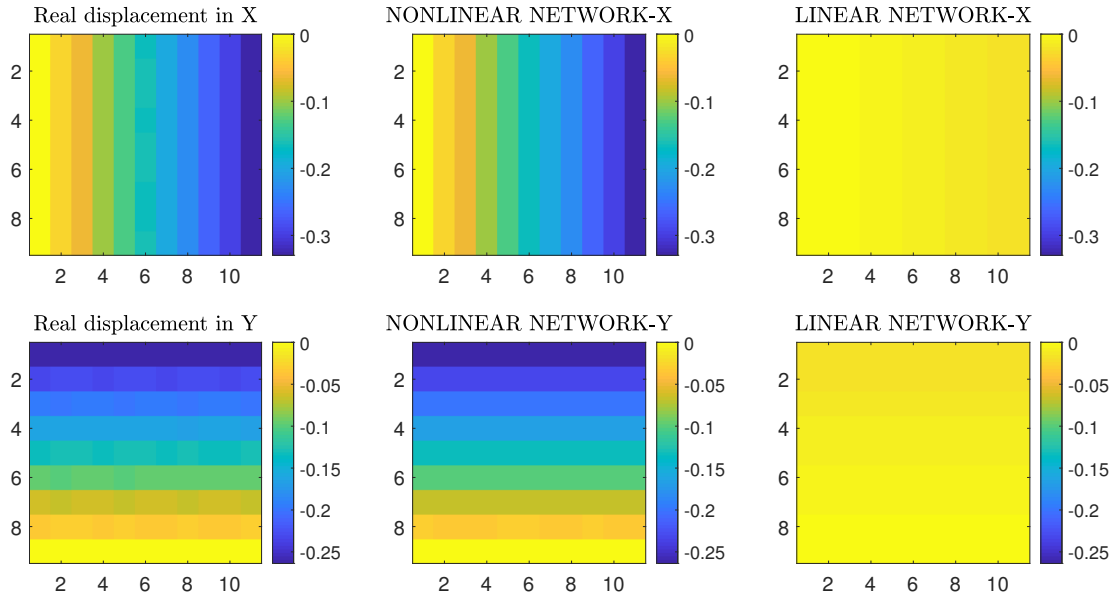


Figure 4.20: Displacement components prediction for the test-based polynomial material (cm).

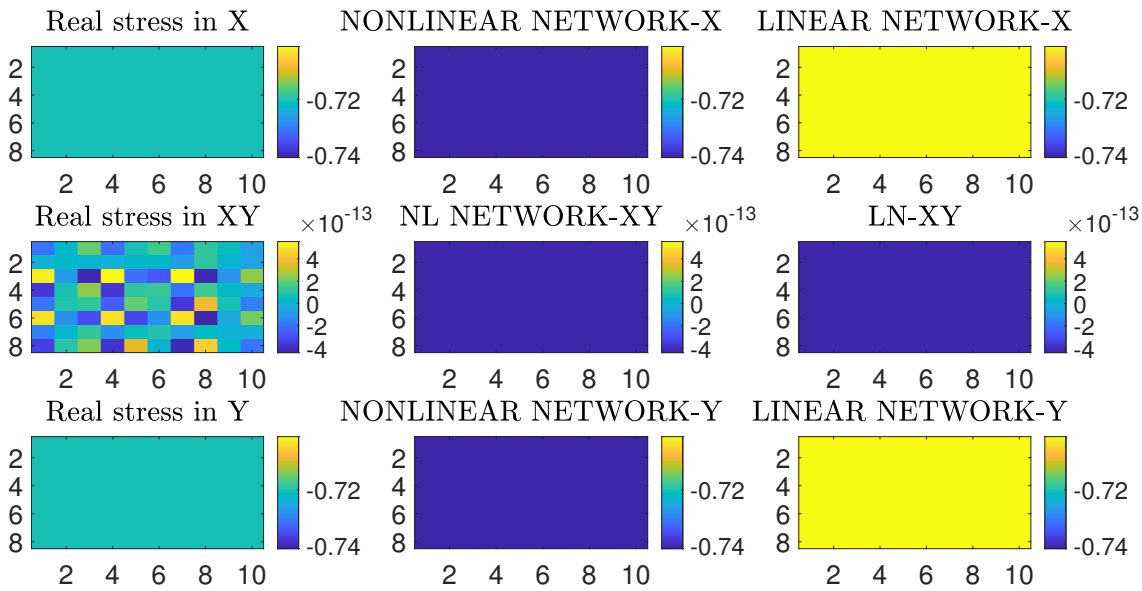


Figure 4.21: Stress components prediction prediction for the test-based polynomial material (MPa).

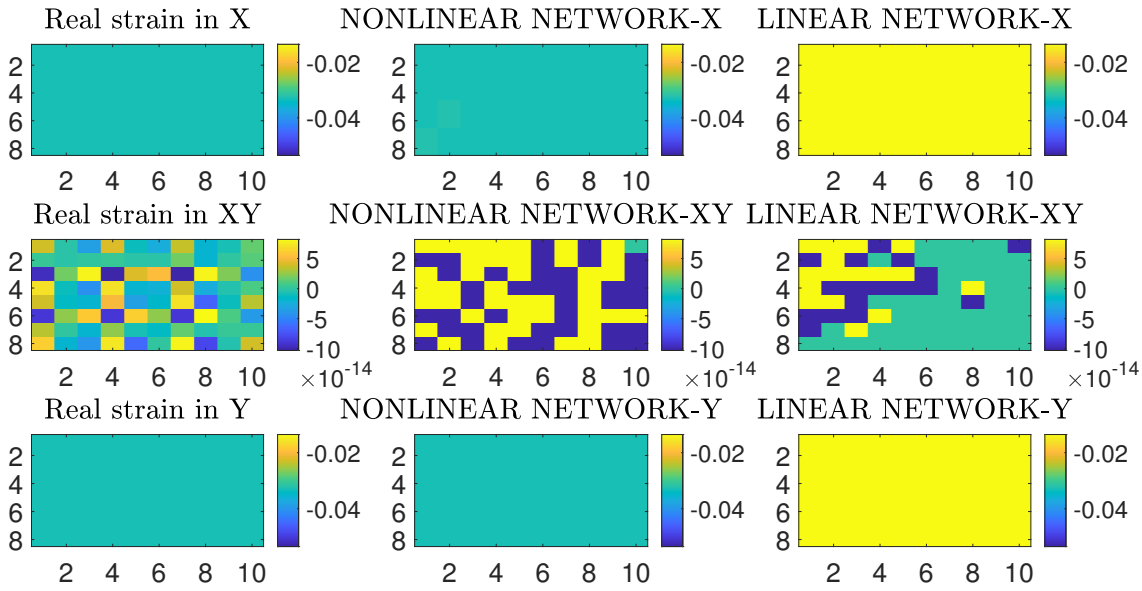


Figure 4.22: Strain components prediction prediction for the test-based polynomial material.

Prediction errors						
	RMSE($\boldsymbol{u}$) (cm)	RMSE($\boldsymbol{\sigma}$) (MPa)	RMSE($\boldsymbol{\varepsilon}$)	RE($\boldsymbol{u}$)	RE($\boldsymbol{\sigma}$)	RE($\boldsymbol{\varepsilon}$)
Nonlinear	2.54×10^{-4}	0.486	5×10^{-3}	0.0018	0.07	0.015
Linear	0.037	0.45	0.09	0.26	0.07	0.26

Table 4.4: Errors for the test-based polynomial material external and internal variables

The predicted errors with the nonlinear PGNNIV for test-based polynomial material presented in Table 4.4 are low, for both displacements and strains. Besides, they are smaller than the ones yielded by the linear network.

In Figures 4.23, 4.24 and 4.25, we present the mean relative error of all variables.

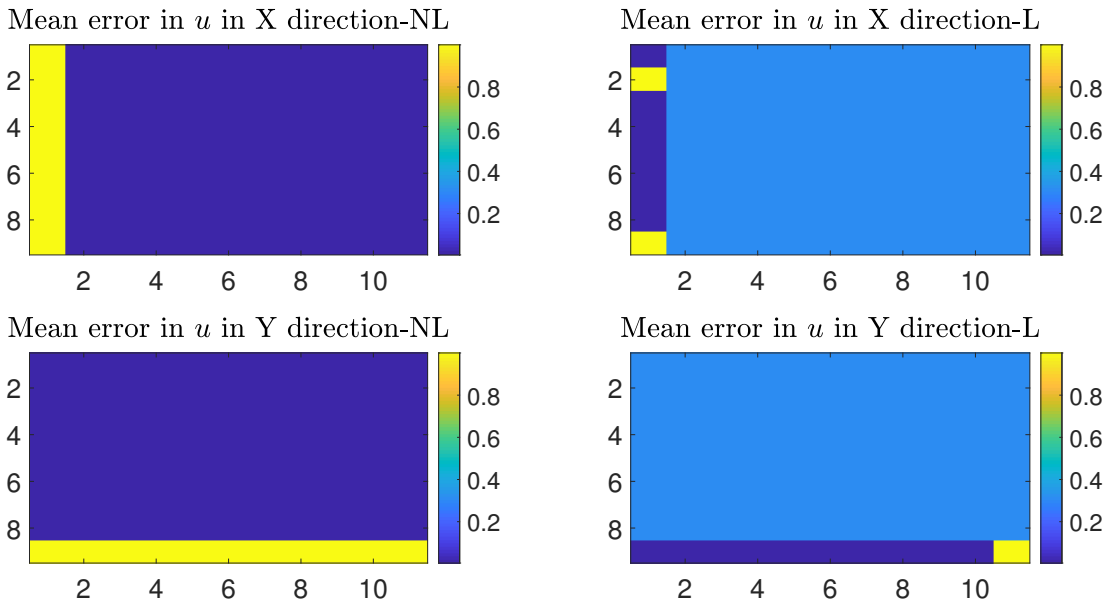


Figure 4.23: Mean relative errorper pixel for the test-based polynomial material (u).

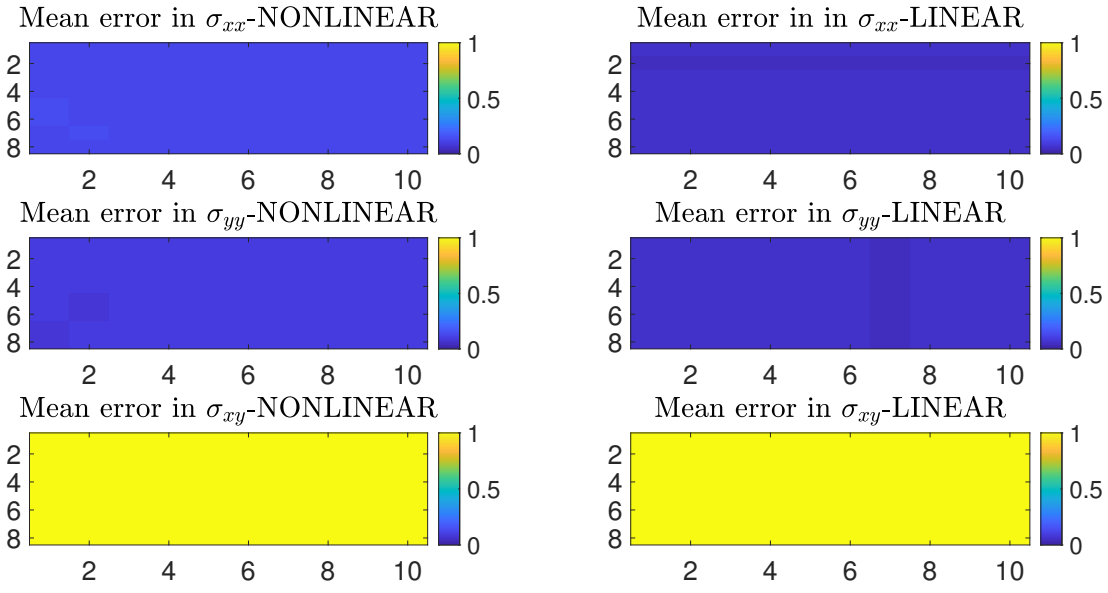


Figure 4.24: Mean relative error per pixel for the test-based polynomial material (σ).

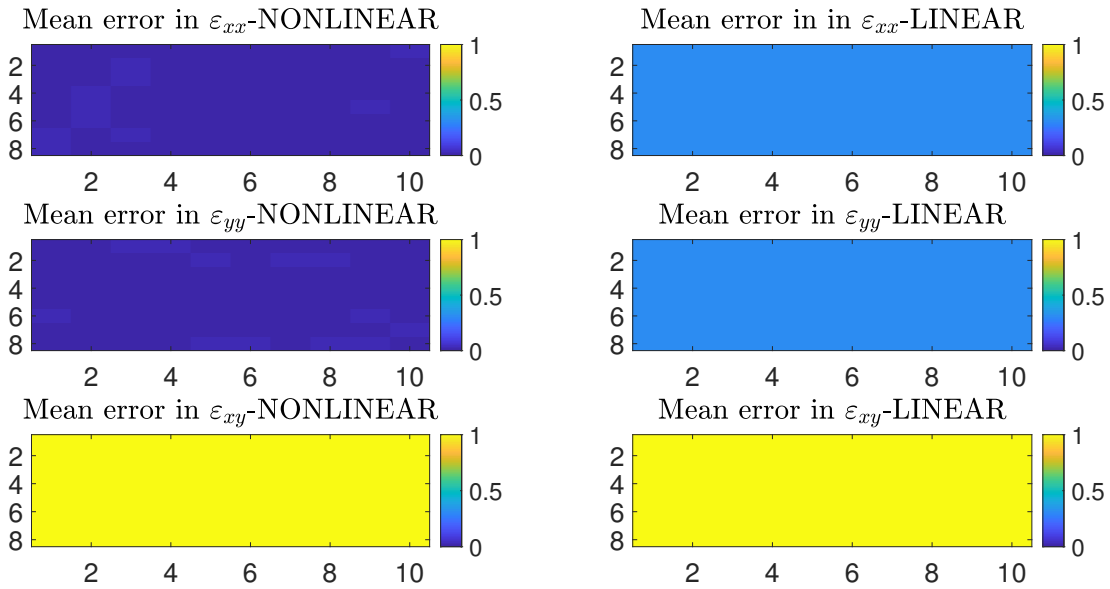


Figure 4.25: Mean relative error per pixel for the test-based polynomial material (ε).

The most important conclusions we can extract are:

– In Figure 4.23 we can observe that the nonlinear prediction of the displacements is accurate. In particular, it is more accurate than the linear prediction. For the strain, the prediction is also accurate.

– In regard to stresses, we have a similar or better prediction of the linear network in all directions. However, these stress fields correspond to an erroneous prediction of the strains. This is a key concept to understand the prediction capacity of the nonlinear PGNNIV. The penalties coefficients p_2 and p_3 (recall Equation (2.35)), associated with the equilibrium of stresses in domain and boundary, are constraining the stress field to be in equilibrium both in the domain and at the boundary. Therefore, the prediction of the stress fields is accurate. However, since the linear PGNNIV sets a fixed relation between stress and strains, these latter are predicted erroneously and it is impossible that the constitutive equation, that is not linear, is correctly unraveled. On the contrary, if p_1 and p_4 had higher values the strain field would be accurately predicted (a solution fulfilling compatibility equations would be found) and the stress field would not, as it would not satisfy equilibrium conditions.

State equation prediction. Figures 4.26, 4.27 and 4.28 show the prediction of both linear and nonlinear networks for the internal variables.

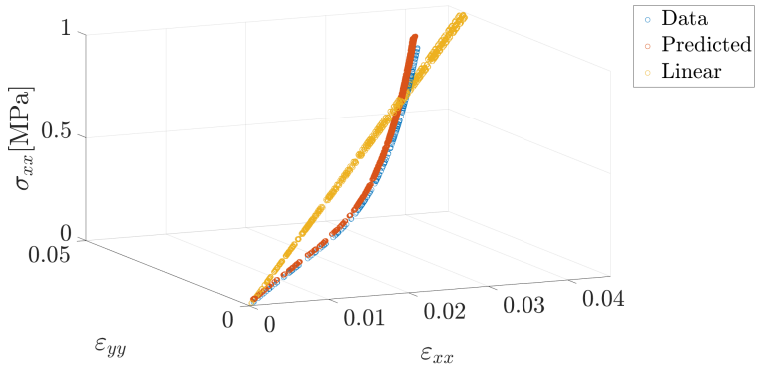


Figure 4.26: Stress prediction σ_{xx} .

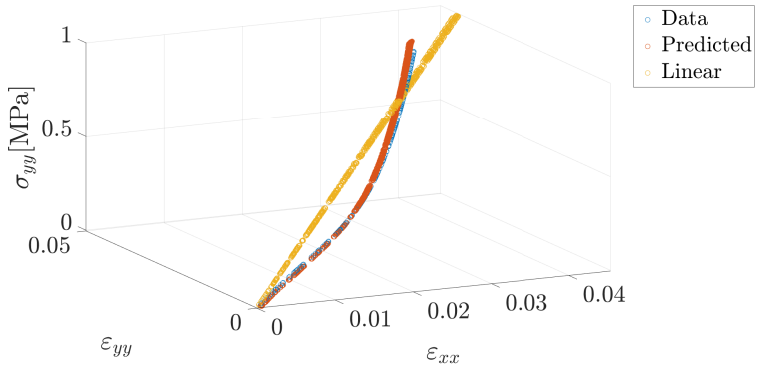


Figure 4.27: Stress prediction σ_{yy}.

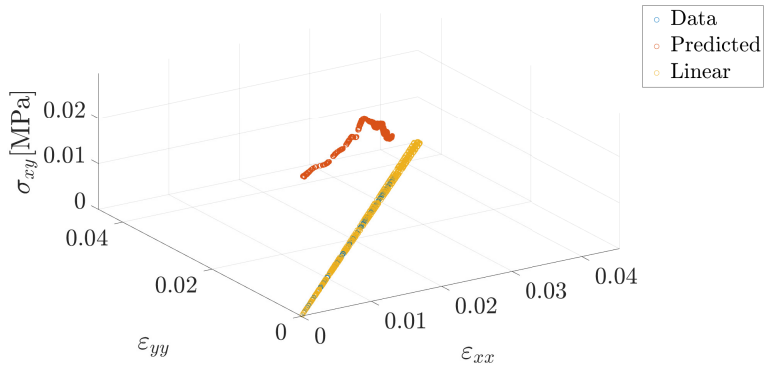


Figure 4.28: Stress prediction σ_{xy}.

As we could have foreseen from Table 4.4 and observe in Figure 4.26 and Figure 4.27, the nonlinear network fitting is better than the linear prediction for σ_{xx} and σ_{yy}. However,

as we can observe in Figure 4.27, σ_{xy}, which is apporximately 0, is predicted more accurately by the linear network. The reason why this occurs is that the linear PGNNIV contains less parameters than the nonlinear PGNNIV. Therefore, the numerical noise that it produces is lower and values close to 0 are predicted more accurately. Figure 4.29 shows the prediction of the strains' and the stresses' values.

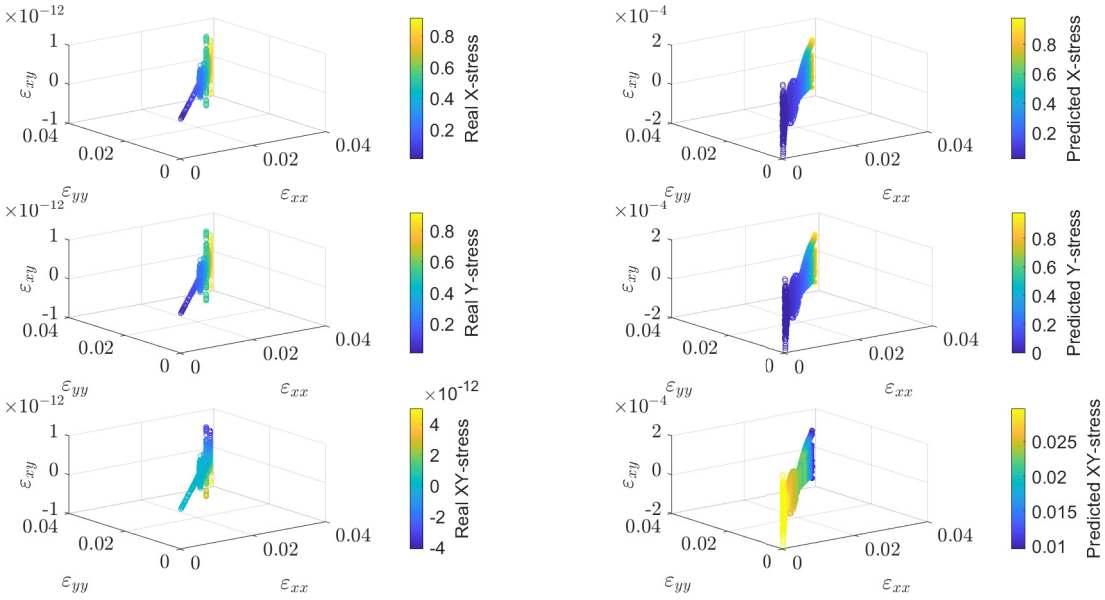


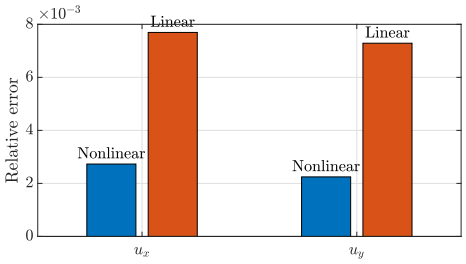
Figure 4.29: Nonlinear PGNNIV strain and stress prediction.

In Figure 4.29, the strain ε_{xy} prediction is by far not accurate, but the stresses, the ε_{xx} and ε_{yy} are accurate. This is due to the fact that the shear stresses and strains of the data-set generated with Abaqus are almost zero, so the network actually learns pure numerical noise, that is dominant for the shear components and no learning capacity is possible.

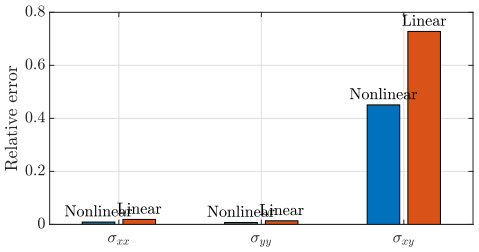
4.4. Summary and Discussion

4.4.1. Predictive capacity

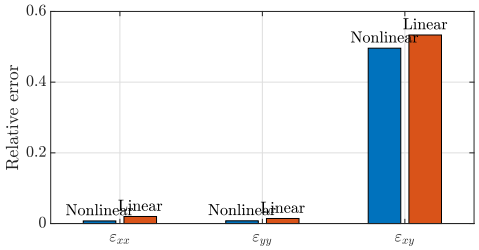
Figures 4.30 and 4.31 show the RE (defined in Equation (3.18)) for both hyperelastic materials respectively.



(a) RE(u) for the Neohookean material.

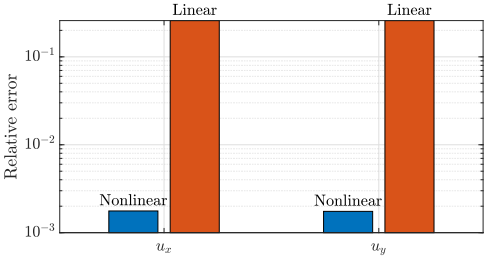


(b) RE(σ) for the Neohookean material.

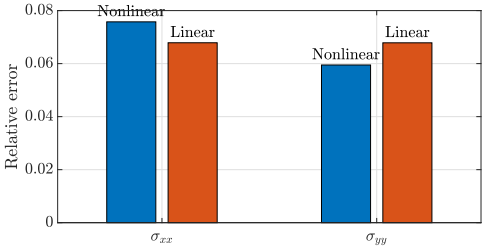


(c) RE(ε) for the Neohookean material.

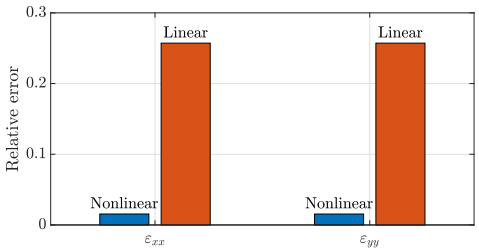
Figure 4.30: Relative global errors for the neohookean material.



(a) RE(u) for the test-based polynomial material.



(b) RE(σ) for the test-based polynomial material.



(c) RE(ε) for the test-based polynomial material.

Figure 4.31: Relative global errors for the test-based polynomial material.

We conclude that:

- For the test-based polynomial material, the developed nonlinear network yields smaller errors for displacements and strains in comparison with the linear network.

- In Figure 4.31b, again for the test-based polynomial material, we can observe that the nonlinear PGNNIV produces a little bit more error than the linear PGNNIV. On the contrary, in Figure 4.31c, we observe that the strains are predicted more accurately by the nonlinear network. This contradiction leads to the fact that the constitutional law is learned accurately by the nonlinear PGNNIV and not by the linear PGNNIV, as we can observe in Figures 4.26, 4.27, 4.28 and 4.29, and is related with what we have yet discussed: the effect of the equilibrium penalties in a better improvement of the prediction of the internal variables.

- The errors are similar between the linear and nonlinear networks for the Neo-hookean material (due to the fact that the latter is highly linear in the chosen range of strains). This indicates us that, at least, from a computational point of view, it is better to work with simpler models (for instance linear models) if we actually know that this is the true behavior of the material.

4.4.2. Explanatory capacity

Should the trained networks be subjected to a uniaxial text, they yield curves in Figures 4.33 and 4.32.

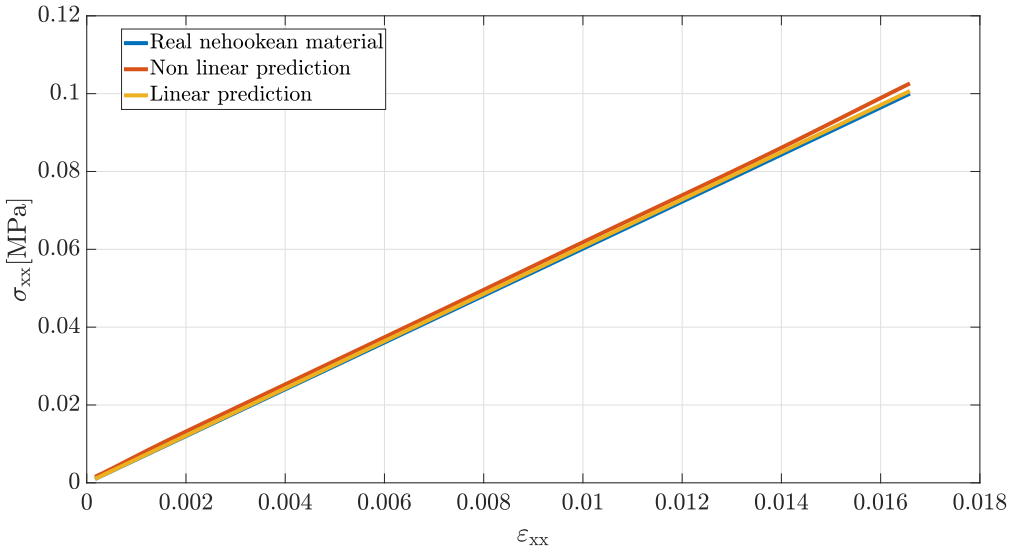


Figure 4.32: $\sigma_{xx} - \varepsilon xx$ curve predicted by the explanatory network for the neohookean material.

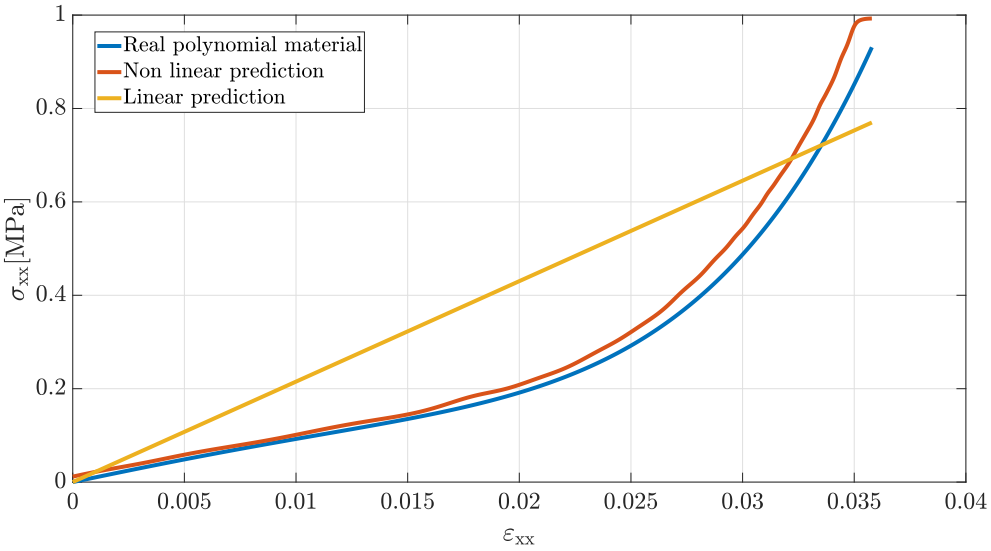


Figure 4.33: $\sigma_{xx} - \varepsilon xx$ curve predicted by the explanatory network for the test-based polynomial material.

As we did in section 3.3.2, we use these curves to calculate the errors between them according to Equation (3.22):

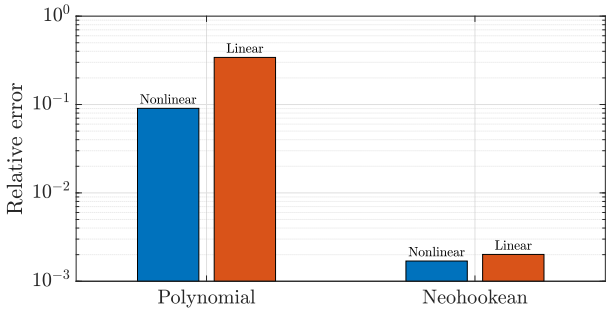


Figure 4.34: Explanatory relative error.

We can observe that the explanatory capacity of the nonlinear network outperforms the one from the linear one, although slightly for the Neohookean material.

4.4.3. Limitations

The limitations of the presented work on hyperelastic materials are multiple. Due to the assumption of the infinitesimal strain theory, the test based material prediction worsens in the last stretch of the curve (see Figure 4.34). A finite strain theory approach might solve therefore this problem. Furthermore, real applications do not consider infinitesimal strains and displacements, so the next step should be developing a network topology based on the finite strain theory.

For simple materials close to the linear behavior, the nonlinear PGNNIV does not improve the results of the linear one. Therefore, before tuning the parameters of a PGNNIV, it is important to know if some assumptions about the nature of the material can be made *a priori*. This results in avoiding possible over-fitting of the numerical noise and in a much less expensive model from the computational point of view.

Furthermore, we only use parabolic profiles as external loads, and the geometry is very limited. There exist multiple types of loads that can act on the solid. Taking them into consideration requires from further development of Equations (2.27). Additionally, in this work, no volumetric loads are considered, and the discretization is fixed. These are all big limitations of our nonlinear PGNNIV model.

Chapter 5

Conclusions and outlook

In this chapter we present the conclusions of this work.

5.1. Summary of the results

To summarize the obtained results during this work even further for each variable $X = \{u_x, u_y, \sigma_x, \sigma_{xy}, \sigma_y, \varepsilon_x, \varepsilon_{xy}, \varepsilon_y\}$ we present the following statistical error definition:

$$\xi_i(X) = \frac{\sqrt{\frac{1}{n_x}\frac{1}{n_y}\sum_{k=1}^{n_x}\sum_{l=1}^{n_y}(\hat{I}^i_{kl}(X) - I^i_{kl}(X))^2}}{\sqrt{\frac{1}{n_x}\frac{1}{n_y}\sum_{k=1}^{n_x}\sum_{l=1}^{n_y}(I^i_{kl}(X))^2}} \tag{5.1}$$

where the sample $i = \{1, \ldots, N\}$, n_x and n_y are the number of pixels in X and Y directions respectively.

The mean of $\xi_i(X)$ is:

$$\mu(X) = \frac{1}{N}\sum_{i=1}^{N}\xi_i(X) \tag{5.2}$$

where N is the size of the test images batch (simulations/samples). The standard deviation of $\xi_i(X)$ is:

$$\sigma(X) = \sqrt{\frac{1}{N}\sum_{i=1}^{N}(\xi_i(X) - \mu(X))^2} \tag{5.3}$$

We can illustrate the variability of the predictions using the statistical error $E(X)$ is therefore:

$$E(X) = \mu(X) \pm \frac{\sigma(X)}{\sqrt{N}} \tag{5.4}$$

This error summarizes the prediction of the nonlinear PGNNIV topology for each material and each variable addressed during the thesis.

Figures 5.1 to 5.6 show the prediction for the X and Y directions for each material, according to the aforementioned error in Equation (5.1).

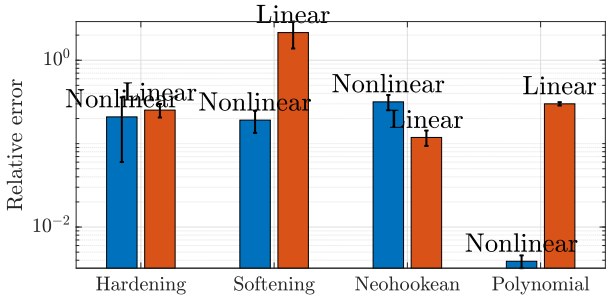


Figure 5.1: Summarized representation of the prediction of u_x.

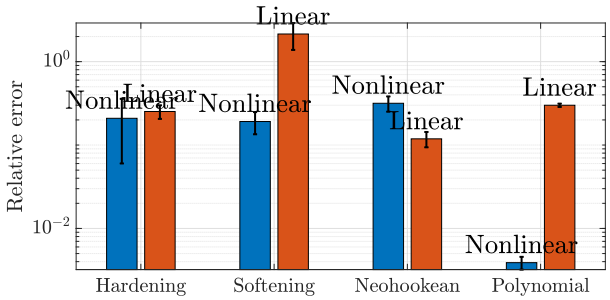


Figure 5.2: Summarized representation of the prediction of u_y.

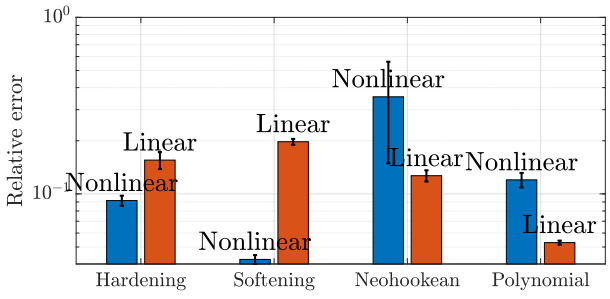


Figure 5.3: Summarized representation of the prediction of σ_{xx}.

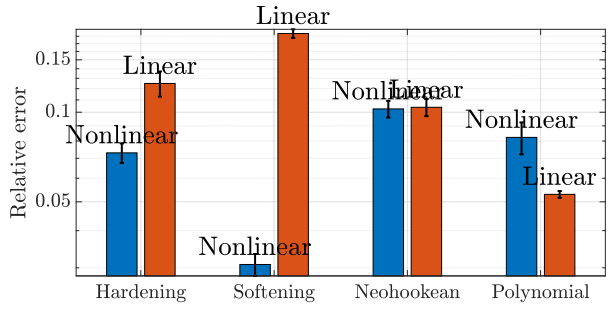


Figure 5.4: Summarized representation of the prediction of σ_{yy}.

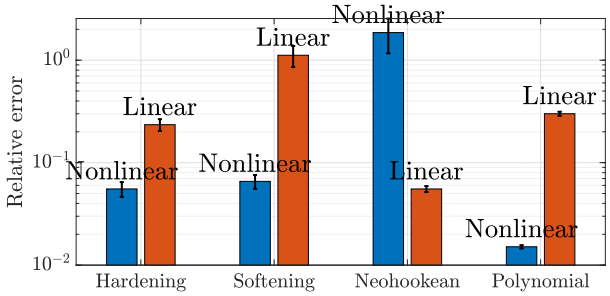


Figure 5.5: Summarized representation of the prediction of ε_{xx}.

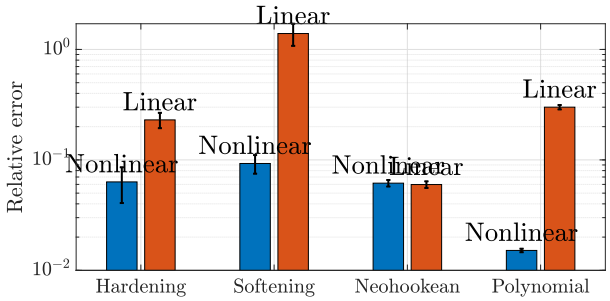


Figure 5.6: Summarized representation of the prediction of ε_{yy}.

In Figures 5.1 to 5.6 we can observe that the nonlinear PGNNIV achieves good estimations for all materials independently from the applied load (the error bar is small), and, except for the Neohookean material, whose predictions are in general similar or worse than the linear one, we notice a better performance of the nonlinear PGNNIV than the linear PGNNIV network. In particular:

– For the prediction of the displacements, and except for the Neo-Hookean case which is *de facto* linear, the nonlinear PGNNIV improves the linear one, in some cases several orders of magnitude.

– For stresses, we have similar results, except for the case of the test-based polynomial material. This is due to the effect induced by the equilibrium penalty, and is therefore an advantage of the PGNNIV method, regardless of it is linear or nonlinear. No method based on classical simulation is be capable of approximating an internal variable such as the stresses, despite having a poorly defined constitutive relationship.

– In the case of strains, the error of the nonlinear PGNNIV is several orders of magnitude lower than that of linear network (except in the case that it is practically linear).

– A general observation is that the error-bar of the nonlinear PGNNIV is larger than the error-bar for linear PGNNIV. This is due to the fact that the nonlinear network inevitably has more parameters and therefore it will incur to some extent into over-fitting, which is induced by the discretization error.

A single nonlinear PGNNIV network topology is, therefore, able to learn the behavior of very different materials. This is a unique feature of PGNNIV and there exist no other methods that can achieve this.

5.2. Main conclusions of the work

Throughout this work, we present the mathematical basis and prediction power of PGNNIV in the field of solid mechanics.

Firstly, the linear PGNNIV topology was analyzed assessing its accuracy when trained with a linear material data-set. Next, the newly developed nonlinear PGNNIV topology was introduced, and innovative topology constructions were designed, so that nonlinear elastic materials with hardening and softening properties can be trained and tested, creating a comprehensive model to unravel their constitutive law out of the imposed boundary conditions and the resulting displacements. The results obtained were satisfactory and the improvement achieved by this new topology in comparison with the linear one was shown by means of different statistical errors and test samples.

Although the nonlinear PGNNIV topology outperformed the linear one for all variables, it did not do it for the internal variables σ_{xy} and ε_{xy}. We concluded therefore that data

from a shear stress loading test had to be incorporated to the training batch in order to achieve more prediction accuracy and a complete description of the material behavior.

Furthermore, we proved that the nonlinear PGNNIV topology was also valid for learning the behavior of hyperelastic and test-based materials. A Neohookean material data-set was used to train the nonlinear PGNNIV, but the results obtained did not outperformed those from the linear network. Since the Neohookean material was very simple and the nonlinear PGNNIV too complex, unnecessary computational resources were deployed, and therefore this led to undesirable over-fitting. Aiming at testing the developed topology with data from a real material, we approximated the test-based constitutive law of a material by a polynomial hyperelastic material. The results achieved for the behavioral prediction under hydrostatic test showed a good performance and high accuracy in the prediction.

All in all, we have proved that PGNNIV applied to a simple solid mechanics case are able to predict important internal and external variables with high accuracy for different nonlinear elastic materials.

However, the presented method has still some limitations:

1. A lot of data is required. In this work, we have generated data synthetically, but in real life sensors are needed. Even so, the inexorable power of Internet of Things can helps us in that direction.

2. Not only a certain (generally large) quantity of data is required, but also its quality is relevant. As we have observed, it is not only important that the data-set is large, but also that it is rich and wide ranging. In this work we have observed that, for example, the lack of sufficiently large shear stimuli made it impossible to learn some features of the material related with the response to these stimuli.

3. Defining a suitable topology for the Y and H networks is not a simple issue and requires from a long process of trial and error, apart from the adjustment of many hyper-parameters. In addition, PGNNIV add extra hyper-parameters, the penalty coefficients, making the process even more difficult and slowing it down.

4. In this work we have worked with relatively small discretizations, but, for larger meshes, neural networks will have more and more parameters and the training stage can become very expensive. Many efforts and research are currently being done on software solutions (distributed computing, parallelization) and hardware (GPUs and TPUs) to accelerate these processes.

5.3. Outlook and future work

The next step the project will be taken to is the formulation of the PGNNIV problem under the finite strain theory framework. For that, we need to define new variables such as the deformation gradient $\boldsymbol{F}$ instead of the Cauchy strain tensor $\boldsymbol{\varepsilon}$, and the Second Piola-Kirchhoff tensor $\boldsymbol{S}$ instead of the Cauchy-stress tensor $\boldsymbol{\sigma}$. Combining these new magnitudes with the already implemented neural network topology, we expect to be able to characterize the behavior of nonlinear elastic and inelastic materials under the aforementioned finite strain theory framework, bringing the PGNNIV concept to a more realistic stage in the current solid mechanics state of the art.

Also, the data-base used for this work was very limited, so creating data-bases corresponding to other materials from synthetic tests in a more systematic and extensive manner poses a big challenge as well.

Furthermore, many other challenges can be tackled. To give some examples, targeting a 3D geometry neural network modeling will still require further algorithmic development and a variable geometry approach should be also pursued in order to analyze real engineering problems. Finally, the presented neural network algorithm in this work was only valid for homogeneous materials, therefore, the creation of new algorithms to learn heterogenoeus materials' behavior still remains as a future challenge.

Finally, in a more advanced stage, applying the PGNNIV approach to real data from sensors, for example, through Digital Image Correlation tests [44], is also one of the future lines.

Bibliography

[1] Daniel T Larose and Chantal D Larose. *Discovering knowledge in data: an introduction to data mining*, volume 4. John Wiley & Sons, 2014.

[2] Mona Tanwar, Reena Duggal, and Sunil Kumar Khatri. Unravelling unstructured data: A wealth of information in big data. In *2015 4th International Conference on Reliability, Infocom Technologies and Optimization (ICRITO)(Trends and Future Directions)*, pages 1–6. IEEE, 2015.

[3] Federico Civerchia, Stefano Bocchino, Claudio Salvadori, Enrico Rossi, Luca Maggiani, and Matteo Petracca. Industrial internet of things monitoring solution for advanced predictive maintenance applications. *Journal of Industrial Information Integration*, 7:4–12, 2017.

[4] Dennis Overbye. Can a Computer Devise a Theory of Everything? . *The New York Times*, 2020.

[5] Jacobo Ayensa-Jiménez, Mohamed H Doweidar, Jose A Sanz-Herrera, and Manuel Doblaré. An unsupervised data completion method for physically-based data-driven models. *Computer Methods in Applied Mechanics and Engineering*, 344:120–143, 2019.

[6] Alexandra L'heureux, Katarina Grolinger, Hany F Elyamany, and Miriam AM Capretz. Machine learning with big data: Challenges and approaches. *Ieee Access*, 5:7776–7797, 2017.

[7] Alex J Smola and Bernhard Schölkopf. A tutorial on support vector regression.

Statistics and computing, 14(3):199–222, 2004.

[8] Aristidis Likas, Nikos Vlassis, and Jakob J Verbeek. The global k-means clustering algorithm. *Pattern recognition*, 36(2):451–461, 2003.

[9] George H Dunteman. *Principal components analysis*. Number 69. Sage, 1989.

[10] Philippos Mordohai and Gérard Medioni. *Manifold Learning*, pages 954–958. Springer US, Boston, MA, 2009.

[11] Yunqian Ma and Yun Fu. *Manifold learning theory and applications*, volume 434. CRC press Boca Raton, 2012.

[12] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[13] Jingying Chen, Ruyi Xu, and Leyuan Liu. Deep peak-neutral difference feature for facial expression recognition. *Multimedia Tools and Applications*, 77(22):29871–29887, 2018.

[14] Long Wen, Xinyu Li, and Liang Gao. A transfer convolutional neural network for fault diagnosis based on resnet-50. *Neural Computing and Applications*, pages 1–14, 2019.

[15] Ayodele Ariyo Adebiyi, Aderemi Oluyinka Adewumi, and Charles Korede Ayo. Comparison of arima and artificial neural networks models for stock price prediction. *Journal of Applied Mathematics*, 2014, 2014.

[16] Adnan Nuhic, Tarik Terzimehic, Thomas Soczka-Guth, Michael Buchholz, and Klaus Dietmayer. Health diagnosis and remaining useful life prognostics of lithium-ion batteries using data-driven methods. *Journal of power sources*, 239:680–688, 2013.

[17] Li C Xue, Drena Dobbs, Alexandre MJJ Bonvin, and Vasant Honavar. Computa-

tional prediction of protein interfaces: A review of data driven methods. *FEBS letters*, 589(23):3516–3526, 2015.

[18] Oliver Lemon and Olivier Pietquin. *Data-driven methods for adaptive spoken dialogue systems: Computational learning for conversational interfaces*. Springer Science & Business Media, 2012.

[19] Rob Kitchin. Big data, new epistemologies and paradigm shifts. *Big data & society*, 1(1):2053951714528481, 2014.

[20] Jiequn Han, Arnulf Jentzen, and E Weinan. Solving high-dimensional partial differential equations using deep learning. *Proceedings of the National Academy of Sciences*, 115(34):8505–8510, 2018.

[21] Quercus Hernández, Alberto Badías, David González, Francisco Chinesta, and Elías Cueto. Structure-preserving neural networks. *Journal of Computational Physics*, 426:109950, 2021.

[22] E. Samaniego, C. Anitescu, S. Goswami, V.M. Nguyen-Thanh, H. Guo, K. Hamdia, X. Zhuang, and T. Rabczuk. An energy approach to the solution of partial differential equations in computational mechanics via machine learning: Concepts, implementation and applications. *Computer Methods in Applied Mechanics and Engineering*, 362:112790, 2020.

[23] Hamid Reza Tamaddon-Jahromi, Neeraj Kavan Chakshu, Igor Sazonov, Llion M. Evans, Hywel Thomas, and Perumal Nithiarasu. Data-driven inverse modelling through neural network (deep learning) and computational heat transfer. *Computer Methods in Applied Mechanics and Engineering*, 369:113217, 2020.

[24] Maziar Raissi, Alireza Yazdani, and George Em Karniadakis. Hidden fluid mechanics: A navier-stokes informed deep learning framework for assimilating flow visualization data. *arXiv preprint arXiv:1808.04327*, 2018.

[25] Jacobo Ayensa-Jiménez, Mohamed H Doweidar, Jose A Sanz-Herrera, and Manuel Doblaré. On the application of physically-guided neural networks with internal

variables to continuum problems. *arXiv preprint arXiv:2011.11376*, 2020.

[26] E Weinan and Bing Yu. The deep ritz method: a deep learning-based numerical algorithm for solving variational problems. *Communications in Mathematics and Statistics*, 6(1):1–12, 2018.

[27] Anuj Karpatne, Gowtham Atluri, James H Faghmous, Michael Steinbach, Arindam Banerjee, Auroop Ganguly, Shashi Shekhar, Nagiza Samatova, and Vipin Kumar. Theory-guided data science: A new paradigm for scientific discovery from data. *IEEE Transactions on knowledge and data engineering*, 29(10):2318–2331, 2017.

[28] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[29] K Karapiperis, L Stainier, M Ortiz, and JE Andrade. Data-driven multiscale modeling in mechanics. *Journal of the Mechanics and Physics of Solids*, 147:104239, 2021.

[30] Ruben Ibañez, Domenico Borzacchiello, Jose Vicente Aguado, Emmanuelle Abisset-Chavanne, Elias Cueto, Pierre Ladeveze, and Francisco Chinesta. Data-driven non-linear elasticity: constitutive manifold construction and problem discretization. *Computational Mechanics*, 60(5):813–826, 2017.

[31] Jacobo Ayensa-Jiménez, Mohamed H Doweidar, Jose A Sanz-Herrera, and Manuel Doblaré. Prediction and identification of physical systems by means of physically-guided neural networks with meaningful internal layers. *Computer Methods in Applied Mechanics and Engineering*, 381:113816, 2021.

[32] Xavier Oliver Olivella and Carlos Agelet de Saracibar. *Mecánica de medios continuos para ingenieros*, volume 92. Univ. Politèc. de Catalunya, 2002.

[33] Manuel Doblare and Luis Gracia. *Fundamentos de Elasticidad Lineal*, volume 11. Sintesis Ed, 1998.

[34] Xavier Oliver and C Agelet de Saracibar. Continuum mechanics for engineers. *Theory and Problems,*, 2017.

[35] MIT Aeroastro, Structural Mechanics. `http://web.mit.edu/16.20/homepage/3_Constitutive/Constitutive.html`. Accessed: 2020-06-10.

[36] Michael A Nielsen. *Neural networks and deep learning*, volume 25. Determination press San Francisco, CA, 2015.

[37] M. Raissi, P. Perdikaris, and G.E. Karniadakis. Physics-informed neural networks: A deep learning framework for solving forward and inverse problems involving nonlinear partial differential equations. *Journal of Computational Physics*, 378:686–707, 2019.

[38] George Papazafeiropoulos, Miguel Muñiz-Calvente, and Emilio Martínez-Pañeda. Abaqus2matlab: A suitable tool for finite element post-processing. *Advances in Engineering Software*, 105:9–16, 2017.

[39] Adam optimizer. `https://www.tensorflow.org/api_docs/python/tf/keras/optimizers/Adam`. Accessed: 2020-06-10.

[40] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014.

[41] Kurt Hornik, Maxwell Stinchcombe, and Halbert White. Multilayer feedforward networks are universal approximators. *Neural networks*, 2(5):359–366, 1989.

[42] Hyperelastic behavior of rubberlike materials . `https://abaqus-docs.mit.edu/2017/English/SIMACAEMATRefMap/simamat-c-hyperelastic.htm`. Accessed: 2020-06-10.

[43] Evaluating hyperelastic and viscoelastic material behavior. `https://abaqus-docs.mit.edu/2017/English/SIMACAECAERefMap/simacae-c-prpeditorevaluate.htm`. Accessed: 2020-06-10.

[44] TC Chu, WF Ranson, and Michael A Sutton. Applications of digital-image-correlation techniques to experimental mechanics. *Experimental mechanics*, 25(3):232–244, 1985.

List of Figures

List of Tables

Appendix

Appendix A

Matlab code for data-set generation by means of co-simulation Matlab-Abaqus.

```
for N_iter=1:NDATA

    % Run Abaqus

    % Abaqus2Matlab: a suitable tool for finite element post-processing.
    % Advances in Engineering Software. Vol. 105, pp 9-16 (2017)
    % DOI:10.1016/j.advengsoft.2017.01.006
    % % G. Papazafeiropoulos, M. Muniz Calvente, E. Martinez-Paneda
    % % % Abaqus2Matlab@gmail.com
    % www.abaqus2matlab.com


    % 1st STEP - Run one FEM model

     % Change the current directory
    S = mfilename('fullpath');
    f = filesep;
    ind=strfind(S,f);
    S1=S(1:ind(end)-1);
    cd(S1) ;

    Inp_file=['Job-',num2str(N_iter)];
    disp('Simulation Started')


```

```
26      % Run the input file with Abaqus
27      % Initialize sw (boolean switch) as true
28      system(['abaqus job=' Inp_file ' interactive']);
29      sw=true;
30      tic;
31      while sw
32      % Pause Matlab execution in order for the lck file to be
        created
33      pause(0.5);
34      % While the lck file exists, pause Matlab execution. If it
        is
35      % deleted, exit the while loop and proceed.
36      while exist([Inp_file '.lck'],'file')==2
37          pause(0.1)
38          % the lck file has been created and Matlab halts in
        this loop.
39          % Set sw to false to break the outer while loop and
        continue
40          % the code execution.
41          sw=false;
42      end
43      % In case that the lck file cannot be detected, then
        terminate
44      % infinite execution of the outer while loop after a
        certain
45      % execution time limit (5 sec)
46      if sw && (toc>5)
47          sw=false;
48      end
49      % NOTE: Alternatively, you can replace lines 27 to 49 by
        system(['abaqus job=' Inp_file ' interactive'])
50      end
51      disp('Simulation Finished')
52
53
54      % 2nd STEP - Postprocess Abaqus results file with
        Abaqu2Matlab
55      % Obtain the desired output data
56      disp('Obtaining desired output data by Abaqus2Matlab')
57      % NOTE: Some output variables are Matlab Cells, If you are
        not confortable working with Cells, you can use cell2mat()
58  if  isfile([ Inp_file '.fil' ])
59
60      U1 =readFil([ Inp_file '.fil' ], 101 );%Obtain the  Nodal
        Displacement(U) ;
61      U=U1{1,1}(end-276:end,1:3);
62
63      S=readFil([ Inp_file '.fil' ],11);%Obtain the Stress(S);
```

```
64      E=readFil([ Inp_file '.fil' ],21);%Obtain the Total Strain
        (E);
65
66  end
```

Appendix B

Abaqus model for nonlinear materials definition and for dataset generation.

B.1. Nonlinear elastic material

Abaqus source code for the hyperelastic material:

```
*Elastic
8000., 0.3
*Plastic
     1e-05,      0.
     0.105,   1e-05
     0.144,   2e-05
     0.172,   3e-05
     0.196,   4e-05
     0.217,   5e-05
     0.235,   6e-05
     0.252,   7e-05
     0.268,   8e-05
     0.283,   9e-05
     0.296,  0.0001
    ......, .......
```

B.2. Nonlinear test-based material

The data used for the definition of the test-based material was introduced in Abaqus, and it writes as follows:

```
*Hyperelastic, n=2, test data input, poisson=0.5

*Biaxial Test Data
 0.009384, 0.0002
   0.0159, 0.0006
 0.024087, 0.0011
  0.02622, 0.0014
  0.03324,  0.002
 0.044278, 0.0031
  0.05183, 0.0042
 0.066024, 0.0068
 0.077794, 0.0094
 0.097857, 0.0149
 0.126351, 0.0203
 0.146804, 0.0243
    0.174, 0.0275
 0.201058, 0.0307
 0.224502, 0.0326
  0.24653, 0.0345

*Planar Test Data
  0.006,  0.00069
  0.016, 0.001034
  0.024, 0.001724
 0.0336, 0.002828
  0.042, 0.004276
   0.06, 0.008483
  0.078, 0.013862
  0.096,     0.02
 0.1112, 0.024897
 0.1296, 0.030345
 0.1488, 0.034483
```

```
 0.1658, 0.037793
  0.182, 0.040621

*Uniaxial Test Data
 0.015506, 0.001338
 0.024367, 0.002675
 0.031013, 0.003567
 0.042089, 0.006242
 0.053165, 0.008917
  0.05981, 0.011592
 0.068671, 0.014268
 0.088608,  0.02051
 0.106329,  0.02586
 0.124051, 0.030318
 0.161709, 0.037898
 0.199367, 0.043694
  0.23481, 0.048153
 0.274684,  0.05172
 0.310127, 0.054395
  0.34557,  0.05707
 0.383228, 0.059299
 0.420886, 0.060637
 0.456329, 0.061975
 0.493987, 0.063312
 0.531646,  0.06465
 0.569304, 0.065541
 0.642405, 0.066433
```

Appendix C

Python code for creating and training a PGNNIV constitutive model for softening, hardening and hyperelastic materials: Global Network.

```
# Force-Displacement Neural network

# Placeholder for input
XTr = tf.placeholder(tf.float32, [None, 1, npxy, 2])
XTt = tf.placeholder(tf.float32, [None, npxx, 1, 2])
Sr = tf.placeholder(tf.float32, [None, npxy-1, 2])
St = tf.placeholder(tf.float32, [None, npxx-1, 2])

print(XTr)
print(XTt)

# Input
X1 = XTr
X2 = XTt

print(X1)
print(X2)


# Output

#Right
B_1=tf.Variable(tf.ones([K_])/100) #100
B_2=tf.Variable(tf.ones([K_])/100)
B__1=tf.Variable(tf.ones([K_])/100)
B__2=tf.Variable(tf.ones([K_])/100)
B__3=tf.Variable(tf.ones([K_])/100)
B__4=tf.Variable(tf.ones([2])/100)

```

```
30
31 Y1=tf.tensordot(X1,W_1,3)
32 Y2=tf.tensordot(X2,W_2,3)
33 print(Y1)
34 Ya=tf.nn.tanh(Y1+Y2)
35 print(Ya)
36 Yb=tf.nn.tanh(tf.tensordot(Ya,W__1,1)+B__1)
37 print(Yb)
38 Yc=tf.nn.tanh(tf.tensordot(Yb,W__2,1)+B__2)
39 Yd=tf.nn.tanh(tf.tensordot(Yc,W__3,1)+B__3)
40 Y=tf.nn.tanh(tf.tensordot(Yd,W__4,1)+B__4)
41
42 # Placeholder for correct output
43 Y_ = tf.placeholder(tf.float32, [None, npxx, npxy,2])
44
45 # Output difference
46 OUT = Y - Y_
47 # Material network
48
49 ## Displacements
50 U = Y
51
52 # Supports
53 UXL = U[:,0,:,0]
54 UYB = U[:,:,0,1]
55
56 # Strains
57 sGRADSU = conv2d(U,WGRADSYM)
58 EPSILON = sGRADSU
59
60 K=50 #number of neurons
61
62 W1=tf.Variable(tf.truncated_normal([3,K],dtype=tf.float32))
63 B1=tf.Variable(tf.ones([K])/10)
64 W2=tf.Variable(tf.truncated_normal([K,K],dtype=tf.float32))
65 B2=tf.Variable(tf.ones([K])/10)
66 W3=tf.Variable(tf.truncated_normal([K,K],dtype=tf.float32))
67 B3=tf.Variable(tf.ones([K])/10)
68 W4=tf.Variable(tf.truncated_normal([K,K],dtype=tf.float32))
69 B4=tf.Variable(tf.ones([K])/10)
70 W5=tf.Variable(tf.truncated_normal([K,K],dtype=tf.float32))
71 B5=tf.Variable(tf.ones([K])/10)
72 W6=tf.Variable(tf.truncated_normal([K,K],dtype=tf.float32))
73 B6=tf.Variable(tf.ones([K])/10)
74 W_FINAL=tf.Variable(tf.truncated_normal([K,3],dtype=tf.float32))
75 B_FINAL=tf.Variable(tf.ones([3])/100)
76
77 SIGMA=[]
78 SIGMA_row=[]
79 for i1 in range(n_xx-1):
80     SIGMA_row=[] #empty new row of SIGMA image
81     for i2 in range(n_xy-1): #loops along the EPSILON pixel image x
    axis
82         EPSILON_pixel=EPSILON[:,i1,i2,:]
83         E1=tf.nn.tanh(tf.matmul(EPSILON_pixel,W1)+B1) #first
    intermediate layer
84         E2=tf.nn.tanh(tf.matmul(E1,W2)+B2) #first intermediate layer
85         E3=tf.nn.tanh(tf.matmul(E2,W3)+B3) #first intermediate layer
```

```
        E4=tf.nn.tanh(tf.matmul(E3,W4)+B4) #first intermediate layer
        E5=tf.nn.tanh(tf.matmul(E4,W5)+B5) #first intermediate layer
        E6=tf.nn.tanh(tf.matmul(E5,W6)+B6) #first intermediate layer
        SIGMA_pixel=tf.nn.tanh(tf.matmul(E6,W_FINAL)+B_FINAL)
        SIGMA_row.append(SIGMA_pixel)
    SIGMA_row=tf.stack(SIGMA_row)
    SIGMA.append(SIGMA_row) #stack a tensor
SIGMA=tf.stack(SIGMA)
SIGMA=tf.transpose(SIGMA, perm=[2,0,1,3])
SIGMA=tf.stack(SIGMA) #stack a tensor



# Unit volume forces
sDIVSIGMA = conv2d(SIGMA,WDIV)
B = sDIVSIGMA


## Equilibrium
# Domain
CONEQD = B
# Boundary



CONEQBr = Sr[:,:,:] - SIGMA[:,9,:,0:2]
CONEQBt = St[:,:,:] - SIGMA[:,:,7,1:3]

#CONEQBl = Tl[:,:,1]
#CONEQBb = Tb[:,:,0]



## Compatibility
# Domain
CONCOMPD = EPSILON - sGRADSU # is identically 0

# Boundary
CONCOMPBx = UXL
CONCOMPBy = UYB

# Initialization
init = tf.global_variables_initializer()

# Loss and error functions

# Squared error
#squared_error = tf.reduce_sum(tf.pow(OUT,2))
error = tf.norm(OUT,axis=None)**2
print(error)
squared_error = error

# Penalty
f1 = tf.norm(CONEQD,axis=None)**2
ff1 = f1

f2 = tf.norm(CONCOMPD,axis=None)**2
ff2 = f2
```

```
144
145 f3r = tf.norm(CONEQBr,axis=None)**2
146 ff3r = f3r
147
148 f3t = tf.norm(CONEQBt,axis=None)**2
149 ff3t = f3t
150
151 # f3b = tf.norm(CONEQBb,axis=None)**2
152 # ff3b = f3b
153
154 # f3l = tf.norm(CONEQBl,axis=None)**2
155 # ff3l = f3l
156
157 f4x = tf.norm(CONCOMPBx,axis=None)**2
158 ff4x = f4x
159
160 f4y = tf.norm(CONCOMPBy,axis=None)**2
161 ff4y = f4y
162
163 pen_fun = pen1*ff1+pen2*ff2+pen3*(ff3r+ff3t)+pen4*(ff4x+ff4y)
164 pen_fun = pen1*ff1+pen3*(ff3r+ff3t)+pen4*(ff4x+ff4y)
165
166
167 # Mean squared error
168 rmse = tf.sqrt(squared_error)
169
170 # Optimizer
171 obj_fun2 = pen0*error + pen_fun
172 #optimizer = tf.train.GradientDescentOptimizer(beta)
173 optimizer = tf.train.AdamOptimizer(beta)
174 train_step2 = optimizer.minimize(obj_fun2)
```

Appendix D

Pixel-wise de-convolution for the H network.

Given an input $\boldsymbol{X}$, consisting on a image of 11×9 pixels, we use weights and biases matrices to scan the image pixel by pixel. Each pixel will be denoted as $\boldsymbol{X}_i$=$[{\varepsilon_{\mathrm{xx}}}^i, {\varepsilon_{\mathrm{xy}}}^i, {\varepsilon_{\mathrm{yy}}}^i]$, where index $i = 2...l$, with l the number of layers. Given a set of weights and biases matrices denoted as $\boldsymbol{W}_i$ and $\mathbf{b}_i$, for each pixel strain component $\boldsymbol{X}_i$ there will be k neurons. The output for each pixel $\boldsymbol{Y}_i = [{\sigma^i}_{\mathrm{xx}}, {\sigma_{\mathrm{xy}}}^i, {\sigma_{\mathrm{yy}}}^i]$, is computed as:

$$\boldsymbol{Y}_i = \phi(\boldsymbol{X}_{i-1}\boldsymbol{W}_i + \boldsymbol{b}_i) \tag{D.1}$$

where $\phi : \mathbb{R}^{n_i} \to \mathbb{R}^{n_i}$ is an hyperbolic tangent activation function.

The image scan programming technique has the following scheme:

Scan imagen decovolutional technique

Data: EPSILON: strain matrix representing an image, EPSILON pixel: strain vector representing a pixel, n_x: number of nodes in X direction, n_y: number of nodes in Y direction, K: number of neurons.

Result: SIGMA: stress matrix representing an image, SIGMA pixel: stress vector representing an image, SIGMA row: Stress vector representing a row of SIGMA

```
n_x=11;
n_y=9;
K=50;
W1= matrix[3,K];
B1= matrix[K];
W2= matrix[K,K];
B2= matrix[K];
W3= matrix[K,K];
B3= matrix[K];
W4= matrix[K,K];
B4= matrix[K];
W5= matrix[K,K];
B5= matrix[K];
W6= matrix[K,K];
B6= matrix[K];
WFINAL= matrix[K,3];
BFINAL= matrix[3];
for i to n_x − 1 do
    SIGMA row=empty[0];
    for i to n_y − 1 do
        EPSILON=EPSILON pixel[:,i1,i2,:];
        E1=tanh(EPSILON*W1)+B1);
        E2=tanh(E1*W2)+B2);
        E3=tanh(E2*W3)+B3);
        E4=tanh(E3*W4)+B4);
        E5=tanh(E4*W5)+B5);
        E6=tanh(E5*W6)+B6);
        SIGMA pixel=tanh((E3*WFINAL)+BFINAL);
        SIGMA row.append(SIGMA pixel);
        SIGMA row=stack(SIGMA row);
        SIGMA.append(SIGMA row);
    end
end
SIGMA=permute([2,0,1,3]);
```

Algorithm 1: Deconvolutional algorithm.